Canada - Ontario Mineral Development Agreement

Entente Canada - Ontario sur l'exploration minérale

Canada

Ontario

COMPUTER PROGRAM SPECIFICATIONS FOR THE ONTARIO MINING INDUSTRY

by

MINING RESOURCE
ENGINEERING LIMITED

CANMET Project No. 6-9009, March 1987

Work on this project was conducted under the auspices of CANMET, Energy, Mines and Resources Canada and funded as part of the Canada - Ontario Mineral Development Agreement.

D.S.S. File Number : 14SQ.23440-6-9009
Contract Serial Number : 23440-6-9009/01SQ
CANMET Scientific Authority : R.F. Boyle

MICROMEDIA

161 pp

TABLE OF CONTENTS

SECTION		PAGE
	LIST OF FIGURES	iv
	LIST OF TABLES	v
	SUMMARY	vi
	SOMMAIRE	vii
	EXECUTIVE SUMMARY	viii
	CHAPTER 1	
1.0	INTRODUCTION	1
	CHAPTER 2 - SOFTWARE ENGINEERING SPECIFICATIONS FOR THE DEVELOPMENT OF SOFTWARE FOR THE MINING INDUSTRY.	
2.0	INTRODUCTION	3
	2.1 Project Size Categories	5
	2.2 Planning the Development Process	5
	2.3 Software Requirements Definition	17
	2.4 Criteria for Software Design	18
	2.5 Criteria for Implementation	25
	2.6 Criteria for Verification	28
	2.7 Software Maintenance	30
	2.8 Supporting Documents	33
	CHAPTER 3 - SPECIFICATIONS FOR THE SELECTION OF OPERATING SYSTEMS	
3.0	INTRODUCTION	35
	3.1 Standard Operating Systems	35
	3.1.1 CP/M	36
	3.1.2 MS-DOS	36
	3.1.3 UNIX	37
	3.2 The Features of UNIX	39
	3.3 UNIX Versus MS-DOS	42

TABLE OF CONTENTS (continued)

SECTION		PAGE
	CHAPTER 4	
4.0	THE SELECTION OF A SINGLE PROGRAMMING LANGUAGE CAPABLE OF CODING ALL MINING RELATED PROGRAMS....	48
	4.1 Language Review.....	52
	4.2 Discussion of Language Review.....	65
	4.3 Langauge Selection.....	71
	CHAPTER 5	
5.0	COMPUTER GRAPHICS.....	73
	5.1 Introduction.....	73
	5.2 The Scope of Computer Graphics.....	73
	5.3 Computer Graphics Applied to the Mining Industry.....	75
	5.4 Issues Concerning the Integration of Graphics in Software in the Ontario Mining Industry.....	78
	5.5 Computer Graphics Technology.....	81
	5.5.1 Hardware.....	81
	5.6 Software.....	85
	5.7 Software Standards.....	87
	5.8 Integration of Software and Hardware.....	91
	5.9 Guideline for the Procurement and Development of Graphics based software in the Ontario Mining Industry.....	92
	CHAPTER 6 - DATABASE SOFTWARE	
6.0	INTRODUCTION.....	99
	6.1 The Database Management System.....	99
	6.2 Components of a Database Management System..	101
	6.3 Specifications for Database Management Systems.....	103
	6.4 Database Administration.....	105
	6.5 The Mining Industry and Database Systems....	105
	6.6 Database Storage Requirements.....	107

TABLE OF CONTENTS (continued)

SECTION		PAGE
	CHAPTER 7	
7.0	COMPUTER HARDWARE REVIEW.....	108
	7.1 The Internal Functioning of a Computer.....	108
	7.2 Computer Storage.....	117
	7.3 Microprocessor and Micro-Computer Selection.	125
	7.4 Input Devices.....	135
	7.5 Output Systems.....	138
	7.6 Modems.....	143
	CHAPTER 8 - SPECIFICATIONS FOR THE CREATION OF A SOFTWARE LIBRARY FOR THE ONTARIO MINING INDUSTRY	
8.0	INTRODUCTION.....	145
	8.1 Mining Software Computer Library.....	145
	CHAPTER 9 - CONCLUSIONS	149
	REFERENCES.....	152

LIST OF FIGURES

FIGURE #		PAGE
1	DISTRIBUTION OF EFFORT IN THE SOFTWARE LIFE CYCLE. (BOEHM, DATAMATION 1973).....	4
2	THE PHASED MODEL OF THE SOFTWARE LIFE CYCLE. (AFTER FAIRLEY).....	7
3	REVIEWS AND MILESTONES IN THE PHASED LIFE CYCLE MODEL. (AFTER FAIRLEY 1985).....	13
4	AN INFORMAL DATA FLOW DIAGRAM OR "BUBBLE CHART"..	20
5	FORMAT OF A STRUCTURE CHART. (FAIRLEY).....	21
6	VISUAL TABLE OF CONTENTS FOR A HIPO PACKAGE. (AFTER FAIRLEY).....	22
7	A HIPO DETAIL DIAGRAM. (AFTER FAIRLEY).....	23
8	A STRUCTURED FLOWCHART AND PSEUDOCODE EQUIVALENT.	24
9	MS-DOS FEATURE EVOLUTION. (AFTER WEINBERG AND GROFF).....	38
10	HISTORY OF UNIX VERSIONS.....	40
11	PC NETWORKS - THE MS-DOS APPROACH.....	45
12	MULTIUSER SYSTEMS - THE XENIX APPROACH.....	46
13	USE OF A DATABASE MANAGEMENT SYSTEM (DBMS).....	100
14	COMPONENTS OF A DATABASE MANAGEMENT SYSTEM. (AFTER SMITH AND BARNES).....	102
15	THE FIVE FUNCTIONAL UNITS OF A COMPUTER SYSTEM...	110
16	THE STORAGE HIERARCHY PYRAMID.....	118
17	THE NUMBER OF BITS THAT CAN BE STORED ON A SILICONE CHIP HAS QUADRUPLED EVERY THREE YEARS, AND THE COST PER BIT OF STORAGE HAS BEEN CUT IN HALF EVERY THREE YEARS. THIS TREND IN SEMI-CONDUCTOR STORAGE IS EXPECTED TO CONTINUE.....	120

LIST OF TABLES

TABLE #		PAGE
1	SIZE CATEGORIES FOR SOFTWARE PRODUCTS.....	6
2	PLANNING A SOFTWARE PROJECT.....	9
3	FORMAT OF A SYSTEM DEFINITION.....	10
4	FORMAT OF A PROJECT PLAN.....	11
5	CONTENT OF AN ARCHITECTURAL DESIGN SPECIFICATION.	15
6	CONTENT OF A DETAILED DESIGN SPECIFICAITON.....	16
7	TYPICAL FORMAT OF SUBPROGRAM AND COMPILATION UNIT PROLOGUES.....	26
8	COMMENTING CONVENTIONS.....	27
9	A COMPARISON OF MS-DOS AND XENIX.....	44
10	COMPARISON OF FIVE PROGRAMMING LANGUAGES.....	56
11	COMPARISON OF HARDWARE CONFIGURATION AND COST....	96
12	A COMPARISON OF SEVERAL 8-, 16-, AND 32-BIT MICROPROCESSORS SHOWING ARCHITECTURAL DIFFERENCES IN THE BUSES, WORD SIZE, AND SPEED.....	129
13	FORMAT OF ABSTRACT REPORT.....	147

SUMMARY

A medium term (3 - 5 year) plan was developed for computer hardware and software utilization by the Ontario Mining Industry. This plan includes software development specifications, the selection of a single operating system for use by mini, micro and mainframe computers and the selection of a single computer language versatile enough to serve all mining related programming needs. Computer hardware and graphics standards were also reviewed.

This review and development of computer hardware and software standards was used to develop a conceptual computer software library for the Ontario Mining Industry. The actual development of this library would eliminate the duplication of programming effort existing at the current time and it would aid in the maintainability of the developed codes.

SOMMAIRE

On a développé un projet à moyen terme (d'une durée de 3 à 5 ans) pour l'exploitation de matériel et de logiciel par l'industrie minière d'Ontario. Ce projet comprend la mise au point des spécifications pour le développement de logiciel d'application, ensuite la sélection d'un système d'exploitation unique à utiliser dans les unités centrales et dans les ordinateurs minis et micros, et enfin le choix d'un langage de programmation qui s'accommode facilement à tous les besoins informatiques qui touchent à l'industrie minière. On a réexaminé les standards en ce qui concerne le matériel et le traitement graphique.

On a utilisé cette révision et ce perfectionnement des standards de matériel et de logiciel pour développer une bibliothèque de concepts logiciels pour l'industrie minière d'Ontario. La réalisation définitive de cette bibliothèque éliminerait la répétition des efforts en programmation qui existe maintenant dans l'industrie et de plus elle contribuerait à la facilité de maintenance des codes déjà développés.

COMPUTER PROGRAM SPECIFICATIONS FOR THE ONTARIO MINING INDUSTRY

- EXECUTIVE SUMMARY -

INTRODUCTION

Many computer programs have been created for Mining Engineering applications by a large number of Universities and companies. Due to the lack of established specifications on developing and maintaining software, the utilization of computer programs has not been maximized.

Mining Resource Engineering Limited has developed a strategy to maximize the use of computer hardware and software by the mining industry. A summary of this work is presented here.

The following subjects have been addressed:

- Specifications for the Development and Maintenance of computer programs
- Selection of a Standard Operating System
- Selection of a Standard Computer Language for Mining Engineering Applications
- Review of Microcomputer Hardware

- Selection of a standard for graphics
- Databases
- The creation of a Mining Software Library to serve the needs of the Industry.

1. Specifications for the Development and Maintenance of Computer Software

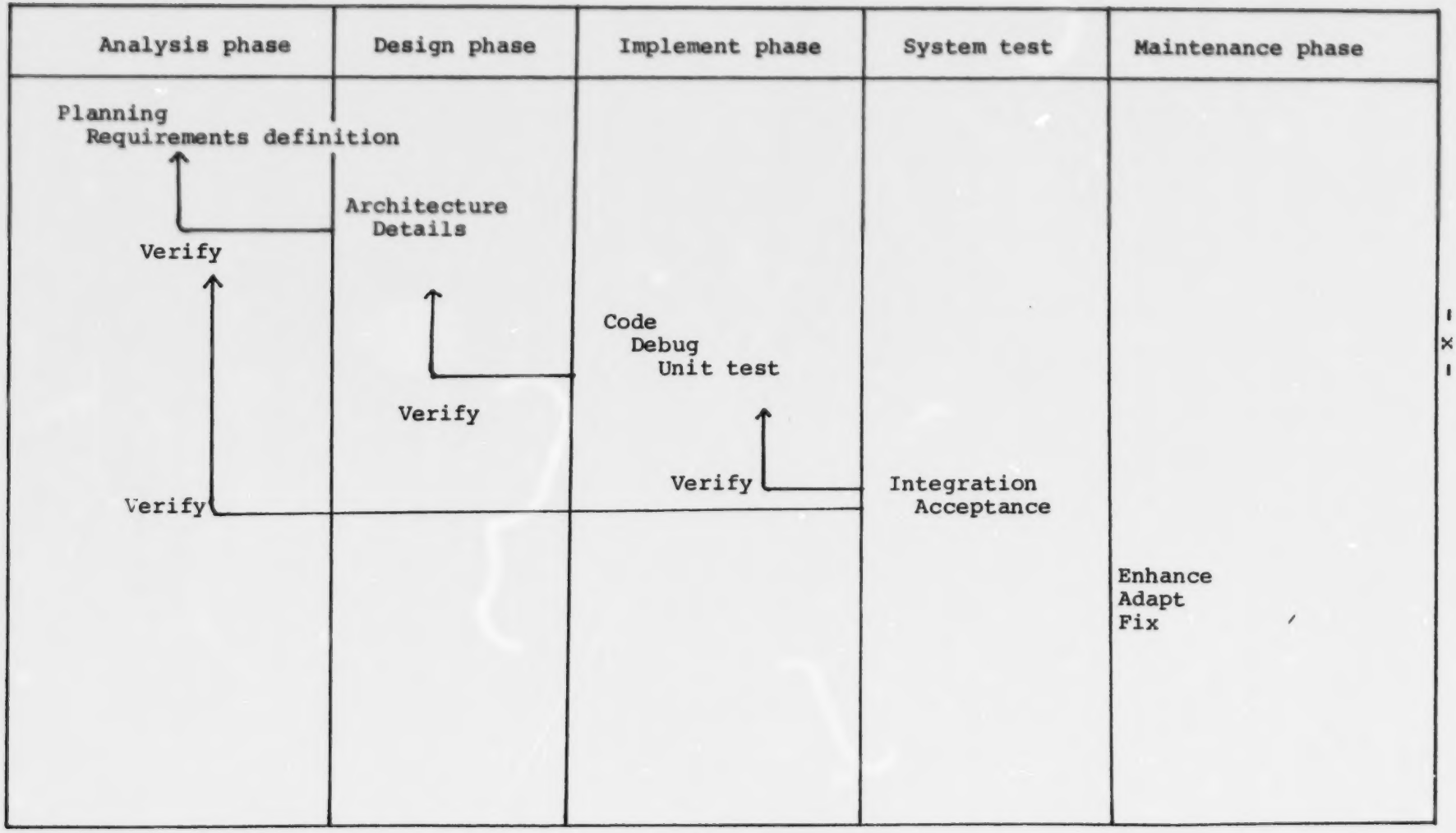
The typical life span for a software product is 1 - 3 years in development and 5 - 15 years in maintenance. The distribution of effort between development and maintenance has been reported as 40/60, 30/70 or even 10/90. Therefore the most important consideration is that the development should enhance the maintainability of the software. In order to achieve this, a product life cycle model has been defined. The software life cycle encompasses all activities required to define, develop, test, deliver, operate and maintain a software product. Figure 1 shows the software life cycle.

Analytical specifications have been set for each of the phases.

The analysis phase deals with planning the software and setting up the requirements for the software.

The design phase should improve the modularity of the code. Design should be done in two stages; architectural and detailed. Standardized notations (structured diagrams, bubble charts and flowcharts) should be used. The design phase should

FIGURE i: THE PHASED MODEL OF THE SOFTWARE LIFE CYCLE



result in a blueprint for the implementation of the code. Moreover the design phase produces the design phase document, which is one of the supporting documents of the code. This document should contain the necessary structured diagrams and detailed flowcharts. The level of detail in the flowcharts should be such, that one notation results in less than 10 lines of coding. For database programs, data flow diagrams should be included.

During implementation the design is translated into code. The following specifications should be followed:

1. Isolate machine dependencies in few subroutines and fully document them.
2. Provide standard documentation prologues for each subprogram.
3. Do not use long subroutines. The maximum number of coding lines should not exceed 50 in each subroutine.
4. Any deviation from the standard should be justified and approved.

The system testing phase of the life cycle includes the verification of the code. The verification must be done in two stages:

1. Static analysis, where inconsistencies in the code (unused variables, uninitialized variables etc.) are detected and eliminated.
2. Dynamic analysis, where the program is executed and the output is evaluated.

The verification phase should result in a report which provides the following information:

1. Type of test.
2. Machine configuration.
3. Test assumptions.
4. Requirements being tested.
5. Expected outcome.
6. Data and results.

The Maintenance phase involves activities which Enhance, Adapt or Fix the code. Maintenance activity can be initiated by a user request or by an error which was detected. The maintenance history of the code must be kept so that the following questions are answered.

1. How many versions of the product exist?
2. How do the versions differ?
3. Which versions are distributed to which sites?
4. What documents are available for each version?
5. What hardware configuration is required to operate a specific version?
6. Which errors were corrected in a specific version?
7. What are the causes of reported product failures?
8. What was the configuration of a given version at a given date?

It is obvious that during the life of the software, review points and standardized documents must be available, because the development process becomes a public activity and the

product becomes more visible and tangible. Reviews are necessary at the end of each of the phases of the presented life cycle model.

After the development of the software the following documents should be available.

1. The design phase document. This document should be given to the customer of the code if the source code is delivered and, therefore, the customer will maintain the code.
2. The theoretical description document. In this document the algorithms used in order to arrive to the solution given are presented.
3. The user's manual. This is the most important document. It should contain the following information.

- Introduction
  - Product rationale and overview
  - Terminology
  - Basic features
  - Outline of the manual
- Getting Started
  - Sign-on
  - Help mode
- Data Files
  - Units and formats
- Modes of operation
- Example runs for each option
- Enhanced features (if any)

microprocessors. Unix has been implemented on systems as small as a personal computer, and on systems as large as the multi-million-dollar Cray II supercomputer.

Unix was selected by this study as the standard operating system for the development of mining engineering applications.

Unix was selected against DOS because of the following features:

- Portability
- Multi-user/Multi-tasking Operation
- Background Processing
- Very large memory addressability
- Advanced Utilities
- Software Development Tools
- Maturity

COMPUTER LANGUAGES

Many programming languages were reviewed and five languages were evaluated in detail. These languages were Basic, C, Fortran, Pascal, and PL/1. The evaluation was based on:

1. Program Structure
2. Statement Layout
3. Data Types
4. Identifiers
5. Declarations
6. Structured Type Array
7. Array Declaration
8. Operators
9. Expressions
10. Assignment

OPERATING SYSTEMS

An operating system is a program which acts as an interface between the user of the computer and the computer hardware.

Standard operating systems emerged in the 1980's as a result of the standardization efforts in the area of microcomputers. The following operating systems were reviewed in detail.

- CP/M
- MS-DOS
- UNIX

CP/M was developed in 1975 for an 8 bit processor. (Intel 8080). However this operating system was quite elementary and soon it was substituted by MS-DOS.

MS-DOS was developed in 1981 and with IBM's support it became the operating system for single user 16-bit personal computers. Although DOS has grown in sophistication since 1981, it is quite limited for scientific and engineering applications. Its main disadvantages are its limited addressability of 640K of CPU memory and its single user and limited multi-tasking capabilities.

UNIX is an "old" operating system. It was developed in the late 1960's by Bell Laboratories and was adapted to microcomputers recently. It's growing popularity is closely related to the development of the 16/32 and 32 bit

11. Conditional and Unconditional Branching
12. Looping/Iteration
13. Function
14. Subroutines
15. Intrinsic Library Functions
16. Input/Output
17. Program Halts
18. Documentation

C was selected as the most desirable language for the multi-purpose programming the mining industry requires. By the 1990's the scope of engineering and scientific programming will have broadened to include robotics control, expert systems coding, machine controlled rock mechanics surveillance, and, many other areas including the traditional "number crunching" type programming. For this broad range of programming, the FORTRAN language will impose severe restrictions and limit programmer efficiency.

C, however, does not restrict the nature of the programming effort. It has been used by systems programmers for coding operating systems and by data base programmers for coding data base programs. It can also be used for a host of other applications such as text processing, machine control and numerical processing.

It must be recognized, however, that many high calibre programs exist in the FORTRAN language. It is <u>not</u> a recommendation of this study to convert these programs to the C language. Rather, only new programs should be coded in the C language.

This introduction of the C language will require both attention and perseverance by mine and government management personnel. Programmers will require C language workshops to become conversant with this language.

In the medium term, the mining industry will benefit immensely from the usage of the C language. The selection of this language allows most mining related programs to be coded in a single language, and increases the portability and uses of each individual program.

COMPUTER HARDWARE

Computer hardware has been examined in considerable detail at the microcomputer level. Areas that have been reviewed include memory, microprocessor components, several microprocessors, microcomputer capabilities and the costs involved for purchase and maintenance of microcomputers.

Microprocessor components have been defined and reviewed. Components such as buses, clock speed, registers, memory management, central processing units have been described in a format that aids in the selection of a microcomputer based on these criteria.

Several microprocessors have been reviewed, with particular emphasis placed on the recently introduced 32 - bit microprocessors. Computers employing these processors should

prove practical for even the most powerful Mining Engineering applications.

Purchase and maintenance costs have been established for various ranges of computing power. The purchase costs range from approximately $2,000.00 to approximately $70,000.00 for workstations. Maintenance costs have been investigated and range from 2 percent to 5 percent of the purchase cost, depending on the environment, in which the computer operates and the distance from service centers.

DATABASE PROGRAMS

A database is a fully integrated collection of files brought together to serve multiple applications. Data is accessed via a database management system which allows the same data to be viewed in a number of different ways.

The Mining Industry is very much interested in database programs. Database systems can be used in the following areas:

- Maintenance
- Inventory
- Accounting
- Drilling - Blasting
- Exploration
- Surveying
- Mine Planning

A database system has the following advantages:

1. More information can be stored in the same amount of space.
2. Less data redundancy is obtained.
3. Conflicting data are eliminated.
4. Data are accessed easily.

However, databases have disadvantages:

- Database systems are more vulnerable than file based systems because of system failures. It is hard to determine the state of a database before failure, especially in multi-user systems.
- Database systems require sophisticated programming and skilled programming personnel.
- The maintenance cost of a database system is high. A system administrator who will look after the database must be employed. In the case of a large system, like the one required in a mining operation, the database administration may require more than two full time positions.

COMPUTER GRAPHICS

Computer graphics has proven itself indispensible in each of the following mining related activities:

- Exploration
- Mine Development
- Mine Production Planning

- Mineral Processing
- Management Support Systems

Several graphics packages have been examined. Today Graphics Kernal System (GKS) is an accepted international standard for the specification of hardware and software for computer graphics. The single largest drawback of GKS is that it is two-dimensional. Work is currently in progress with ANSI and ISO on the implementation of three-dimensional primitives to GKS.

Microcomputers have in the past been overlooked for graphics applications due to their inferior graphics capabilities. However, with the emergence of graphics processors such as Intel's 80786 Graphics Coprocessor and Texas Instruments' 34010 Graphics System Processor, microcomputer capabilities have been drastically expanded.

Graphics based hardware has been reviewed. This hardware includes input devices such as: graphics tablets, joysticks, the mouse, light pens and output devices such as printers, plotters, and cameras.

MINING SOFTWARE LIBRARY

In order to maximize the usage of available software, it is desirable that a library of computer software for the Mining Industry is created. The Mining Software Library should serve the following purposes:

1. Information on Existing Software
2. Distribution of Software
3. Maintenance of Software

The first step towards the creation of a Library should be the collection of available software and literature related to the software packages. Since the already existing programs do not follow the exact software specifications established by the present work, it is recommended that each program is tested, evaluated and an evaluation report is prepared. It is desirable that a copy of each program is available for sole testing by the Library staff. If this is not possible, the program will be evaluated at any site which will provide a working up to date version of the code.

Moreover the library will compile a list of abstracts for the programs available to the Mining Industry. The abstract will have the format of Figure 2.

The Library will also try to obtain software which will be distributed and maintained through it. This can be done in conjunction with software developers and Universities. In this case a maintenance report and proper documentation will be available for each code supported by the Library.

MEMBERSHIP ISSUES

In order for the effort to be viable, it has to be sponsored by the Mining Industry. Therefore a membership fee

will be required. It is suggested that the following policy is adopted:

- Canadian Mining Companies will provide an initial fee which will give them the right to obtain evaluation reports and software for as long as the funds last. Upon exhaustion of the funds the fee should be renewed.
- Canadian Mining Schools will have access to the Library at no cost, provided that the Library will have their cooperation and support.
- Foreign companies and Universities will be able to become members at an increased fee.

CONCLUSIONS

Software engineering principles concerning the production and maintenance of software products for the use of the mining industry have been developed. It is realized that a software product developed for the mining industry has many users and maintainers. Therefore, it is important that computer programs are developed on the concepts of ease of maintenance and use. A software product consists of several deliverable items in addition to the source code. Typical deliverables include a user's manual, the design phase document and the theoretical description document. Specifications were set for each of these documents.

Standard operating systems were examined for use by mining companies. UNIX was selected as a standard because of its portability, multi-user, multi-tasking capabilities and its memory addressability.

A large variety of computer languages have been examined in order to select the most appropriate one for mining engineering applications. The programming needs of the mining industry are extensive and include database and text processing programs, traditional numerical processing programs and low level programming for such activities as robotics and machine control. Five languages (BASIC, C, FORTRAN, PASCAL, and PL/1), were evaluated. The C language was selected because of its multi-purpose capabilities. C is an excellent general purpose language and its programming style leads to portable computer programs.

Graphics software and hardware for use by the mining industry have been reviewed in considerable detail. For 3 dimensional graphics, the Graphical Kernel System (GKS) has not been internationally standardized as of yet and it may be several years before a standard is set. For this reason several three dimensional packages have been examined.

Database management systems have been reviewed. Problems associated with databases and advantages from their use were presented and the applications for the mining industry were indicated.

Microcomputer hardware has been examined. This consisted of a review of the functioning of the microcomputer with emphasis placed on the understanding of microcomputer specifications. Computer hardware including input\output devices, memory and modems were examined. Rule of thumb costs were developed for a range of microcomputer systems and support hardware for computer maintenance.

Finally a library of mining related software was proposed. This library will collect information on existing software and distribute and maintain software products. It will also produce evaluation reports and complete abstracts for available software.

1. NAME AND DESTINATION OF THE PROGRAM.
2. COMPUTER FOR WHICH PROGRAM IS DESIGNED AND OTHER MACHINE PACKAGES AVAILABLE.
3. DESCRIPTION OF PROBLEM.
4. DESCRIPTION OF SOLUTION.
5. RESTRICTIONS ON THE COMPLEXITY OF THE PROBLEM.
6. TYPICAL RUNNING TIME.
7. TYPICAL PRIMARY AND SECONDARY MEMORY REQUIREMENTS.
8. UNUSUAL FEATURES OF THE PROGRAM.
9. PROGRAMMING LANGUAGES USED.
10. OPERATING SYSTEM UNDER WHICH PROGRAM IS EXECUTED.
11. PROGRAMMING OR OPERATING INFORMATION OR RESTRICTIONS.
12. STATUS.
13. AUTHOR(S).
14. AVAILABILITY.
15. MATERIAL AVAILABLE.
16. SUPPORT BY THE AUTHOR.
17. SUPPORT BY THE LIBRARY.
18. RELATED AND AUXILIARY PROGRAMS.
19. REFERENCES.

FIGURE ii: FORMAT OF THE ABSTRACT REPORT.

CHAPTER 1

1.0 INTRODUCTION

The computer has been used by the mining industry for numerical modeling for many years. Many specialized programs have been created for engineering related purposes by large mining companies and consultants. These programs have generally been developed using one computer and the computer language selected for the coding has generally been selected on the basis of language familiarity. The use of non-universal computers, languages and operating systems both limits the portability and increases the costs associated with moving a program from one environment to another.

The purpose of this contract was to develop a medium term plan (3 - 5 year) for computer hardware and software utilization by the Ontario mining industry. The plan has been subdivided into seven primary areas:

1. development of software specifications,
2. evaluation and selection of an operating system,
3. the review and selection of a universal programming language for all mining related activities,
4. review of the use of graphics in mining and the selection of a graphics standard,
5. review of mining related data bases,
6. review of computer hardware and the development of rule of thumb costs for various microcomputer systems.

7. the conceptual design of a computer software library based on the standards selected.

By enforcing the standards selected and developed in this medium term plan, and through the development of the software library as is proposed, the existing and newly created computer programs will be made available to the mining industry through a single management body. Data sheets and program summaries would be made available to all members of the library. In this fashion the mining industry will maximize the industrial utility of engineering and scientific programs.

CHAPTER 2

SOFTWARE ENGINEERING SPECIFICATIONS FOR THE DEVELOPMENT OF SOFTWARE FOR THE MINING INDUSTRY

2.0 INTRODUCTION

The typical lifespan for a software product is 1 - 3 years in development and 5 - 15 years in maintenance. The distribution of effort between development and maintenance has been reported to vary between 40/60 and 30/70 and in some circumstances, 10/90. This is not surprising, when it is understood that maintenance comprises all activities following initial release of a software product. Software maintenance involves three types of activities: enhancing the capabilities of the product, adapting the products to new processing environments and correcting errors. The distribution of effort in the software life cycle is shown in figure 1.

It is obvious that a strategy for software development should be directed towards the following objectives:
First, to increase quality and productivity during initial development, and second, to assist in systematic testing, enhancement, adaptation and correction of the product.

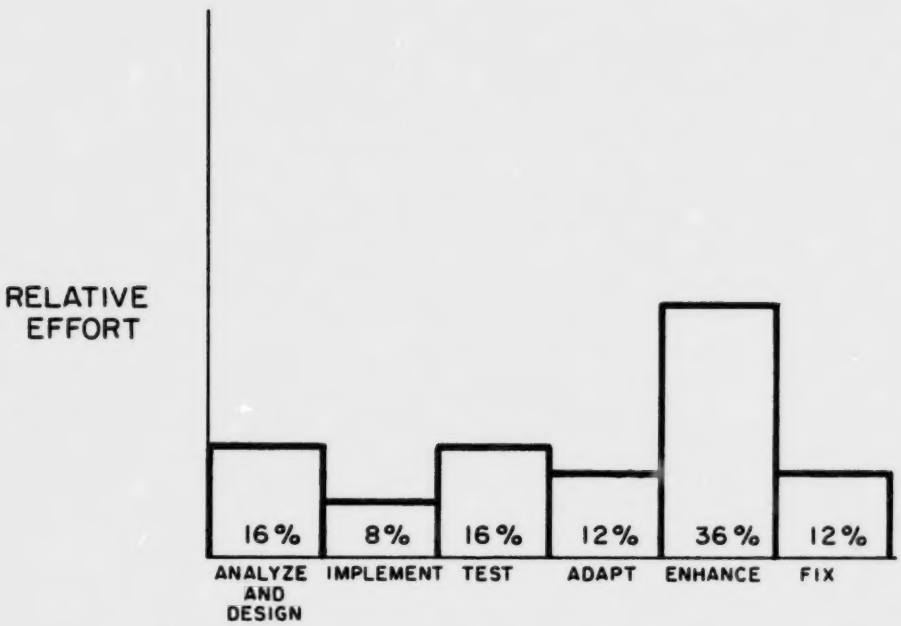


FIGURE 1 : DISTRIBUTION OF EFFORT IN THE SOFTWARE LIFE CYCLE (BOEHM, DATAMATION 1973).

2.1 Project Size Categories

Project size is a major factor that determines the level of management control and the types of tools and techniques required on a software project. The categories shown in table 1, give some indication of size.

It is obvious that the mining industry is mostly interested in small to medium size programs, with a few exceptions towards the large programs when dealing with large finite element analysis codes. The need for formal analysis, elaborate design documentation, extensive test planning and supporting documents becomes more important with the program size. Nevertheless, formal planning is beneficial to any product which is intended to exceed the limits of the personal use. In the following, the formal planning of a software project will be analyzed.

2.2 Planning the Development Process

The first consideration is to define a product life cycle model. The software life cycle encompasses all activities required to define, develop, test, deliver, operate and maintain a software product. A life cycle model best suited to the needs of the mining industry is shown in figure 2.

Analysis consists of two phases: planning and defining the software requirements. The activities during

CATEGORY	NUMBER OF PROGRAMMERS	DURATION	PRODUCT SIZE
Trivial	1	1-4 wks.	500 source lines
Small	1	1-6 mos.	1K - 2K
Medium	2-5	1-2 yrs.	5K - 50K
Large	5-20	2-3 yrs.	50K - 100K
Very Large	100-1000	4-5 yrs.	1M
Extremely Large	2000-5000	5-10 yrs.	1M - 10M

TABLE 1: SIZE CATEGORIES FOR SOFTWARE PRODUCTS.

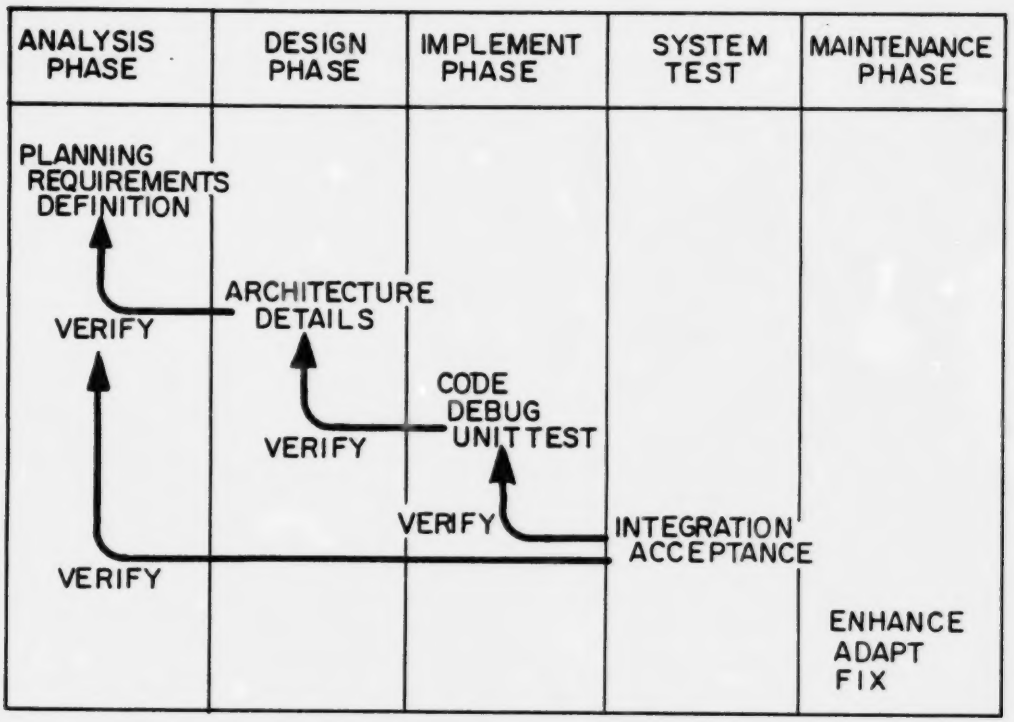


FIGURE 2 : THE PHASED MODEL OF THE SOFTWARE LIFE CYCLE.
AFTER FAIRLEY

planning are summarized in table 2. They include understanding of the problem, performing a feasibility study, developing a recommended solution strategy, determining the acceptance criteria and planning the development process. The outcomes of planning are a System Definition and a Project Plan. The contents of these outcomes are shown in tables 3 and 4.

The definition of software requirements is concerned with identifying the basic functions of the software component in the system hardware/software/people. Here the interest is on what the software is to do and the constraints under which it will perform its functions. Deciding exactly how the software will be implemented is deferred until the design phase. However, the processing environment, the required software functions, performance constraints and acceptance criteria for the software have to be defined.

The next phase is the design phase. Design consists of architectural design and detailed design. Architectural design involves identifying the software components, decoupling and decomposing them into processing modules and specifying the interconnections between them.

Detailed design is concerned with the details of how to package the processing modules and how to implement the processing algorithms. It involves adaptation of existing code, modification of standard algorithms and invention of new algorithms.

Defining the Problem

1. Develop a definitive statement of the problem to be solved. Include a description of the present situation, problem constraints, and a statement of the goals to be achieved. The problem statement should be phrased in the customer's terminology.

2. Justify a computerized solution strategy for the problem.

3. Identify the functions to be provided by, and the constraints on, the hardware subsystem, the software subsystem, and the people subsystem.

4. Determine system-level goals and requirements for the development process and the work products.

5. Establish acceptance criteria for the system.

Developing a Solution Strategy

6. Outline several solution strategies.

7. Conduct a feasibility study for each strategy.

8. Recommend a solution strategy, indicating why other strategies were rejected.

9. Develop a list of priorities for product characteristics.

Planning the Development Process

10. Define a life-cycle model and an organizational structure for the project.

11. Plan the configuration management, quality assurance, and validation activities.

12. Establish preliminary cost estimates for system development.

13. Establish a preliminary development schedule.

14. Establish preliminary staffing estimates.

15. Develop preliminary estimates of the computing resources required to operate and maintain the system.

16. Identify sources of information, and refer to them throughout the project plan.

TABLE 2: PLANNING A SOFTWARE PROJECT.

SECTION 1:	PROBLEM DEFINITION
SECTION 2:	SYSTEM JUSTIFICATION
SECTION 3:	GOALS FOR THE SYSTEM AND THE PROJECT
SECTION 4:	CONSTRAINTS ON THE SYSTEM AND THE PROJECT
SECTION 5:	FUNCTIONS TO BE PROVIDED (HARDWARE/SOFTWARE/PEOPLE)
SECTION 6:	USER CHARACTERISTICS
SECTION 7:	DEVELOPMENT/OPERATING/MAINTENANCE ENVIRONMENTS
SECTION 8:	SOLUTION STRATEGY
SECTION 9:	PRIORITIES FOR SYSTEM FEATURES
SECTION 10:	SYSTEM ACCEPTANCE CRITERIA
SECTION 11:	SOURCES OF INFORMATION
SECTION 12:	GLOSSARY OF TERMS

TABLE 3: FORMAT OF A SYSTEM DEFINITION.

SECTION 1:	LIFE-CYCLE MODEL TERMINOLOGY/MILESTONES/WORK PRODUCTS
SECTION 2:	ORGANIZATION STRUCTURE MANAGEMENT STRUCTURE/TEAM STRUCTURE/WORK BREAKDOWN STRUCTURE/STATEMENTS OF WORK
SECTION 3:	PRELIMINARY STAFFING AND RESOURCE REQUIREMENTS STAFFING AND RESOURCE SCHEDULE
SECTION 4:	PRELIMINARY DEVELOPMENT SCHEDULE
SECTION 5:	PRELIMINARY COST ESTIMATE
SECTION 6:	PROJECT MONITORING AND CONTROL MECHANISMS
SECTION 7:	TOOLS AND TECHNIQUES TO BE USED
SECTION 8:	PROGRAMMING LANGUAGES
SECTION 9:	TESTING REQUIREMENTS
SECTION 10:	SUPPORTING DOCUMENTS REQUIRED
SECTION 11:	MANNER OF DEMONSTRATION AND DELIVERY
SECTION 12:	INSTALLATION PLAN
SECTION 13:	MAINTENANCE CONSIDERATIONS
SECTION 14:	METHOD AND TIME OF DELIVERY
SECTION 15:	METHOD AND TIME OF PAYMENT
SECTION 16:	SOURCES OF INFORMATION

TABLE 4: FORMAT OF A PROJECT PLAN

In the end of the design phase, a blueprint should be provided for the implementation phase.

The implementation phase involves the translation of design specifications into source code, debugging, documentation and unit testing of the source code.

System testing involves two kinds of testing: integration and acceptance. Integration testing ensures that the modules can be integrated into one working code. Acceptance testing requires various tests in order to demonstrate that the implemented software system satisfies the requirements stated in the requirements document.

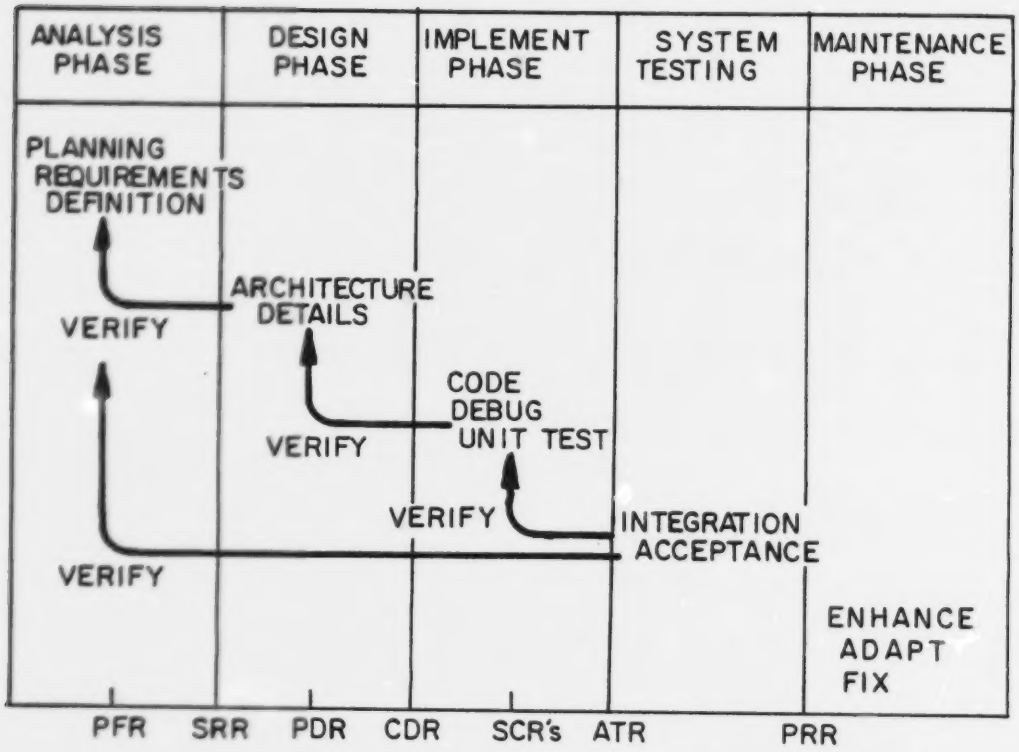
Maintenance activities include enhancement of capabilities, adaptation to new processing environments and correction of software errors.

During the process of the previous phase diagram, milestones, documents and reviews will be generated. This is illustrated in figure 3.

Milestones, review points and standardized documents are essential, because the development process becomes a more public activity and the product becomes visible and tangible.

In the following the work products and the reviews of them are explained:

1. A system Definition and Project Plan are prepared using the formats of tables 3 and 4. A product feasibility review is then held to determine the feasibility of project



REVIEW	WORK PRODUCTS REVIEWED
PFR : PRODUCT FEASIBILITY REVIEW	SYSTEM DEFINITION PROJECT PLAN
SRR : SOFTWARE REQUIREMENTS REVIEW	SOFTWARE REQUIREMENT SPECIFICATION
PDR : PRELIMINARY DESIGN REVIEW	ARCHITECTURAL DESIGN DOCUMENT
CDR : CRITICAL DESIGN REVIEW	DETAILED DESIGN SPECIFICATIONS USER'S MANUAL
SCR : SOURCE CODE REVIEW	WALKTHROUGHS & INSPECTIONS OF THE SOURCE CODE
ATR : ACCEPTANCE TEST REVIEW	ACCEPTANCE TEST PLAN
PRR : PRODUCT RELEASE REVIEW	ALL OF THE ABOVE

FIGURE 3 : REVIEWS AND MILESTONES IN THE PHASED LIFE-CYCLE MODEL. AFTER FAIRLEY 1985

continuation. The outcome might be termination, redirection or continuation of the project.

2. A preliminary user's manual which provides a method of communication between customer and developer is prepared.
3. A Software Requirements Specification is prepared.
4. A Software Requirements Review is held to ensure the adequacy of the System. The goal of the requirements review is to assure that everyone agrees on terminology, everyone interprets the requirements the same way and to expose problem areas.
5. The Software Design Specification is done in two stages: architectural and detail. The contents of the two stages are shown in tables 5 and 6.
6. A Preliminary Design Review is held to evaluate the adequacy of the architectural design in satisfying the Software Product Specification. More than one review might be necessary.
7. After the completion of the detailed design, a Critical Design Review is held. At the same time the preliminary user's manual is updated.
8. Implementation starts. During implementation Source Code Reviews are held to ensure that all code has been reviewed by at least one person other than the programmer.
9. The User's Manual is completed during the implementation phase.

- DATA FLOW DIAGRAMS FOR THE SOFTWARE PRODUCT
- CONCEPTUAL LAYOUT OF DATA STRUCTURES AND DATABASES
- NAMES, DIMENSIONAL UNITS, AND OTHER ATTRIBUTES OF DATA OBJECTS
- NAME AND FUNCTIONAL DESCRIPTION OF EACH MODULE
- INTERFACE SPECIFICATIONS FOR EACH MODULE
- INTERCONNECTION STRUCTURE OF THE MODULES
- INTERCONNECTIONS AMONG MODULES AND DATA STRUCTURES
- TIMING CONSTRAINTS
- EXCEPTION CONDITIONS

TABLE 5: CONTENT OF AN ARCHITECTURAL DESIGN SPECIFICATION.

- PHYSICAL LAYOUT OF DATA STRUCTURES AND DATABASES
- DATA DICTIONARY SPECIFICATION OF ALL CONCRETE DATA OBJECTS
- DETAILED ALGORITHMS FOR EACH NEW MODULE TO BE WRITTEN
- ADAPTATIONS REQUIRED FOR EXISTING CODE THAT WILL BE REUSED
- SPECIFIC PROGRAMMING TECHNIQUES REQUIRED
- INITIALIZATION PROCEDURES
- PACKAGING OF MODULES INTO THE PHYSICAL IMPLEMENTATION

TABLE 6: CONTENT OF A DETAILED DESIGN SPECIFICATION.

10. Prior to delivery a Final Acceptance Review is performed. The final review confirms that all requirements specified in the requirement specifications have been met, the source code and all external documents are complete, consistent and ready for delivery.

2.3 Software Requirements Definition

Requirements specified during initial planning are made specific in order to define the features which the software product will incorporate. The requirements specification will state the "what" of the software product.

The following topics need to be addressed:

1. Operational environments. The hardware system, the operating system and the implementation language has to be defined.
2. Memory constraints. The memory range should be sufficiently wide to allow the programmer some freedom.
3. Performance specifications.
4. Implementation priorities.
5. Acceptance criteria. The standards to be applied to source code, internal documentation and external documents, as well as the method of verifying the product.
6. Accuracy of the product.

2.4 Criteria for Software Design

Design involves conceiving, planning and specifying the internal structure and the processing details of the software product. The design is concerned with refining the conceptual view of the system, identifying internal processing functions, decomposing high level functions into subfunctions, and establishing interconnections between functions and data. During detail design, algorithms are specified and the packaging scheme for the system is established. The fundamental concepts for the design of software packages are:

1. Abstraction
2. Information hiding
3. Structure
4. Modularity
5. Ease of modification
6. Aesthetics

The above concepts are difficult to quantify and standardize. However, the result of design must be well defined representation schemes. The following design notations may be used:

1. Data Flow Diagrams

Data flow diagrams ("bubble charts") are directional graphs in which the nodes specify processing activities and the arcs specify data items transmitted between processing nodes. Data flow diagrams can be used at any level of

abstraction and they do not indicate decision logic. An example of a data flow diagram is presented in figure 4.

2. Structure Charts

Structure charts must be used in the architectural design to document hierarchical structure, parameters and interconnections in a system. A structure chart is shown in figure 5.

3. Hierarchy - Process - Input - Output (HIPO) diagrams

A set of HIPO diagrams contains a visual table of contents, a set of overview diagrams and a set of detailed diagrams. The visual table of contents is a directory to the set of diagrams in the package. It consists of a tree structured directory, a summary of contents of each overview diagram and a legend of symbol definitions. Overview diagrams describe input, output and processing steps and can point to other detail diagrams. The nature of a HIPO diagram is shown in figures 6 and 7.

4. Pseudocode

Pseudocode can be used at any level of abstraction. The system characteristics are described by short phrases that are structured by key words.

5. Flowcharts

These are the traditional means of specifying the algorithmic details of a system. An example of a flowchart is shown in figure 8.

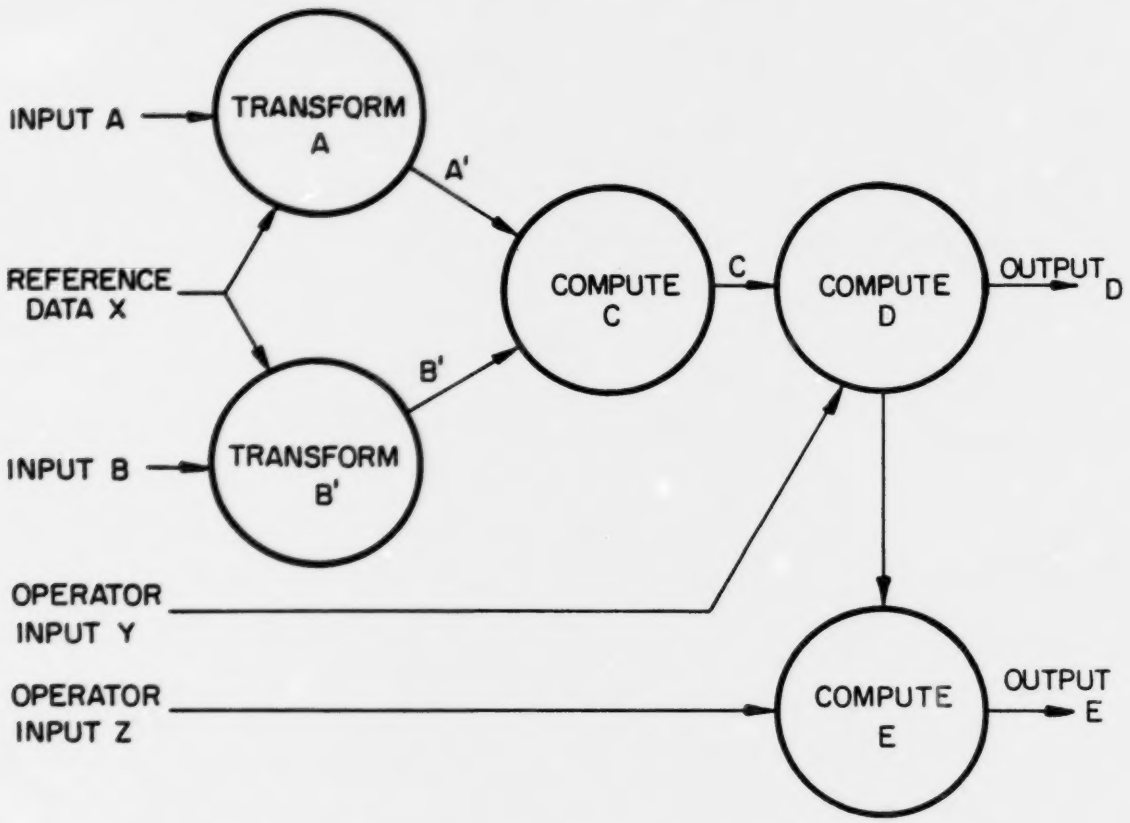
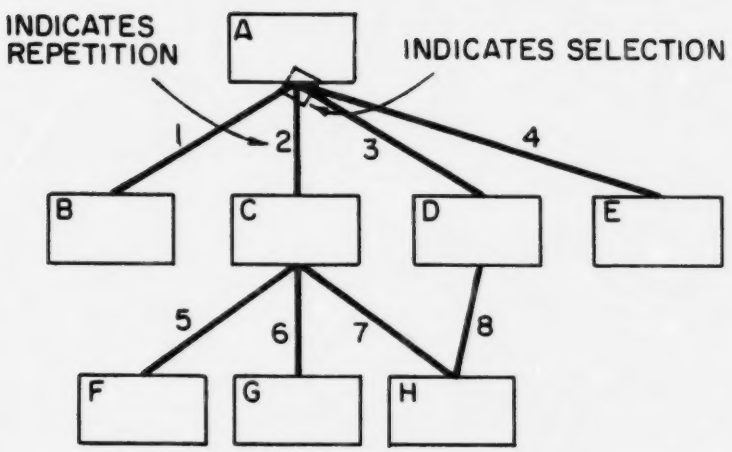


FIGURE 4 : AN INFORMAL DATA FLOW DIAGRAM OR "BUBBLE CHART."



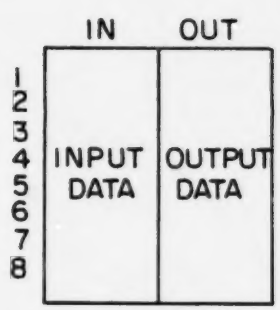


FIGURE 5 : FORMAT OF A STRUCTURE CHART . (FAIRLEY)

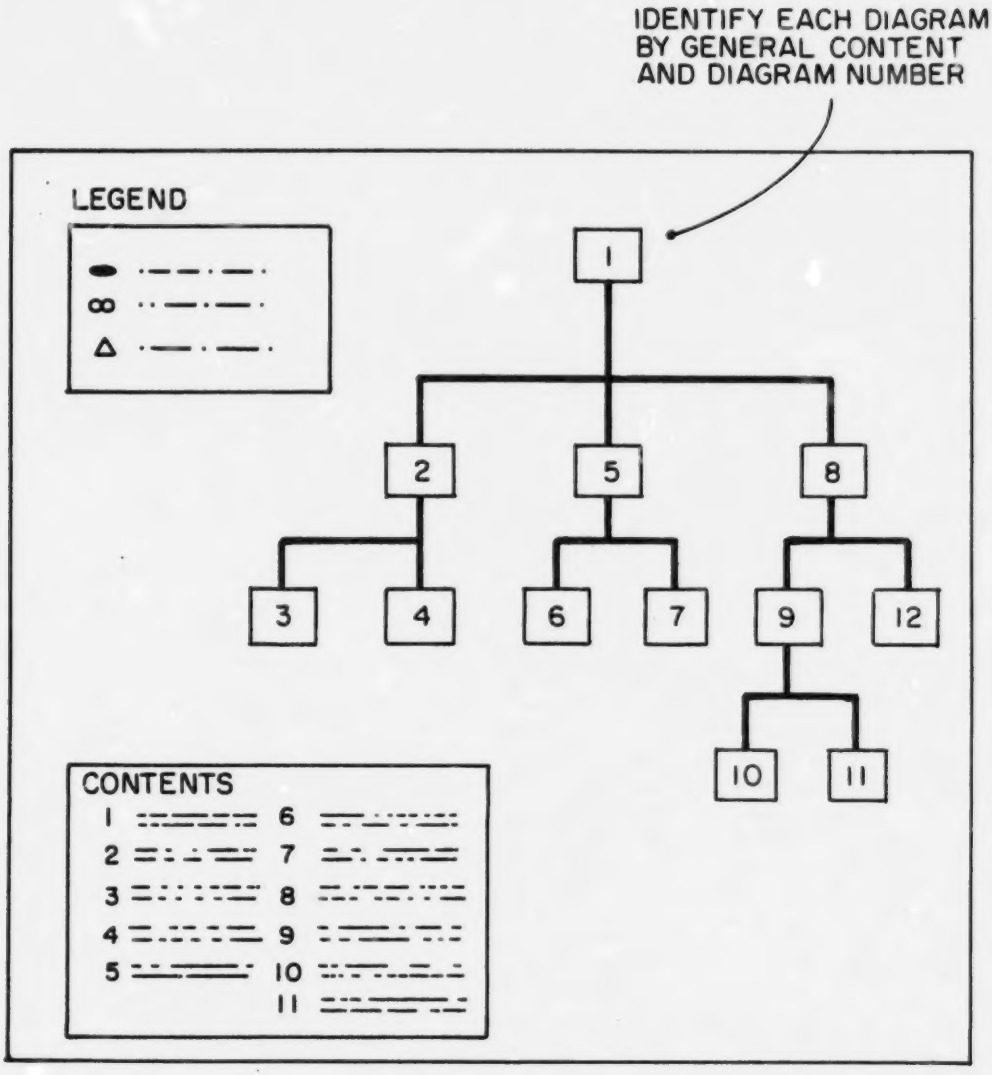


FIGURE 6 : VISUAL TABLE OF CONTENTS FOR A HIPO PACKAGE.
AFTER FAIRLEY

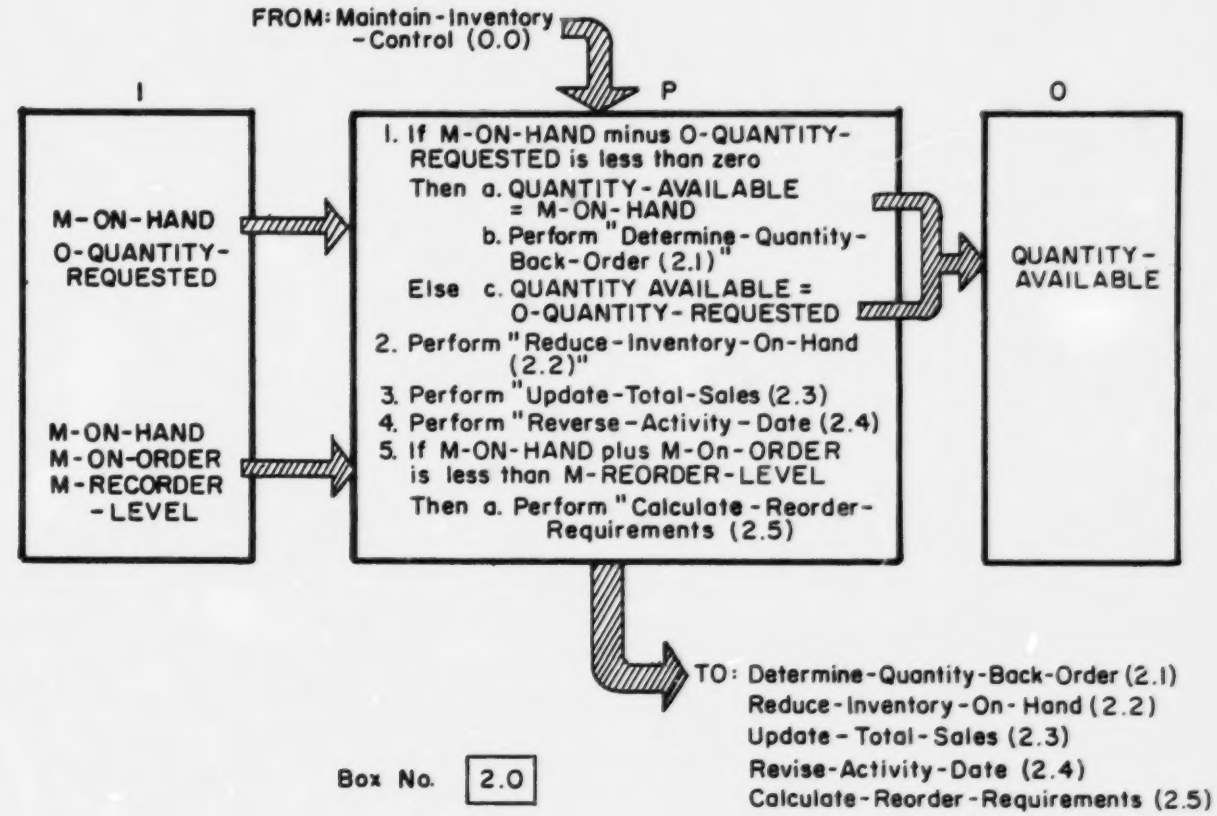


FIGURE 7 : A HIPO DETAIL DIAGRAM. AFTER FAIRLEY

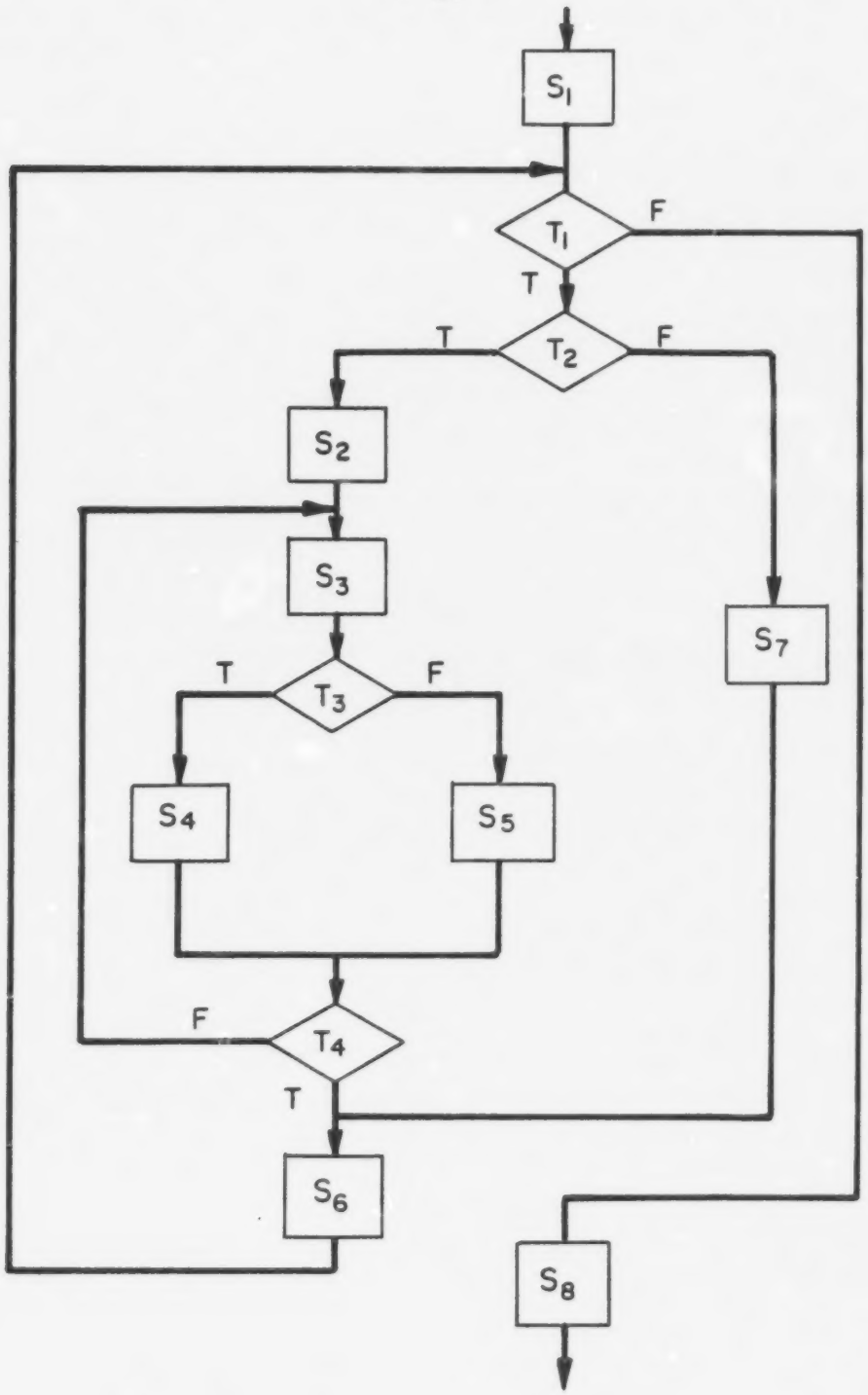


FIGURE 8 : A STRUCTURED FLOWCHART AND PSEUDOCODE EQUIVALENT.

From the above notations, structure charts and detailed flow charts are necessary for mining scientific applications and must be included in the supporting documents of the package.

Moreover, the level of detail in the detailed flowcharts should be such, that each design notation will result in less than ten statements of implementation language.

2.5 Criteria for Implementation

The implementation is concerned with translating design specifications into source code. The code should be as clear and straightforward as possible.

The dynamic structure of the program as it executes should resemble the static structure of the written text.

In the following, guidelines for coding are presented:

1. Isolate machine dependencies in few routines and document them. By doing so, when the software is moved to a different machine, details of machine dependencies are localized and can be modified accordingly with the minimum effort.
2. Provide standard documentation prologues for each subprogram. A typical format for prologues is shown in table 7.
3. Provide internal comments, embedded in the executable portion of the code, to enhance its readability. The conventions which should be followed are shown in table 8.
4. Use meaningful variable and identifier names.

NAME OF AUTHOR:

DATE OF COMPILATION:

FUNCTION(S) PERFORMED:

ALGORITHMS USED:

AUTHOR/DATE/PURPOSE OF MODIFICATIONS:

PARAMETERS AND MODES:

INPUT ASSERTION:

OUTPUT ASSERTION:

GLOBAL VARIABLES:

SIDE EFFECTS:

MAJOR DATA STRUCTURES:

CALLING ROUTINES:

CALLED ROUTINES:

EXCEPTION HANDLING:

ASSUMPTIONS:

TABLE 7: TYPICAL FORMAT OF SUBPROGRAM AND COMPILATION UNIT PROLOGUES.

1. MINIMIZE THE NEED FOR EMBEDDED COMMENTS BY USING:
   - STANDARD PROLOGUES
   - STRUCTURED PROGRAMMING CONSTRUCTS
   - GOOD CODING STYLE
   - DESCRIPTIVE NAMES FROM THE PROBLEM DOMAIN FOR USER-DEFINED DATA TYPES, VARIABLES, FORMAL PARAMETERS, ENUMERATION LITERALS, SUBPROGRAMS, FILES, ETC.
   - SELF-DOCUMENTING FEATURES OF THE IMPLEMENTATION LANGUAGE,
2. ATTACH COMMENTS TO BLOCKS OF CODE THAT:
   - PERFORM MAJOR DATA MANIPULATIONS
   - SIMULATE STRUCTURED CONTROL CONSTRUCTS USING GOTO STATEMENTS
   - PERFORM EXCEPTION HANDLING
3. USE PROBLEM DOMAIN TERMINOLOGY IN THE COMMENTS.
4. USE BLANK LINES, BORDERS, AND INDENTATION TO HIGHLIGHT COMMENTS.
5. PLACE COMMENTS TO THE FAR RIGHT TO DOCUMENT CHANGES AND REVISIONS.
6. DON'T USE LONG, INVOLVED COMMENTS TO DOCUMENT OBSCURE, COMPLEX CODE. REWRITE THE CODE.
7. ALWAYS BE SURE THAT COMMENTS AND CODE AGREE WITH ONE ANOTHER, AND WITH THE REQUIREMENTS AND DESIGN SPECIFICATIONS.

TABLE 8: COMMENTING CONVENTIONS.

5. The number of executable statements in a subprogram should not be more than 40 in normal cases. This would enhance the understanding and the readability of the code.

2.6 Criteria for Verification

The verification of the code should be done in two stages; static and dynamic. In the static analysis stage the code is not executed. However the structure of the code is examined and structural errors, (uninitialized variables, departures from coding standards) are located. The static analysis will provide the following information:

Control flow graph.

Symbol table for variables (including statement numbers where defined, set, and used).

Call graph (subprograms called by each routine).

Arguments passed to each routine; where called from.

Uninitialized variables

Variables set but not used

Isolated code segments

Departures from coding standards (language standards, project standards).

Alignment errors in COMMON blocks (FORTRAN).

Misuse of subprogram parameters: incorrect number of actual parameters, mismatched types between actual and formal parameters, formal output parameters not set by called routine, formal output parameters set but not used by calling routine, actual parameters not used for input or output by called routine.

Total number of lines.

Total number and location of comment lines.

Total number and location of blank lines.

Number and location of labels and goto statements.

Of course for a code to be acceptable it will have to pass the static analysis test and any departures from standards should be justified.

During the dynamic analysis stage the code should be executed. The tests for the acceptance of the code should be planned in conjunction with the customer.

Each test should be formally documented and the following should be specified:

1. Type of test.
2. Machine configuration.
3. Test assumptions.
4. Requirements being tested.
5. Expected outcome.

The data and results should become part of the formal documentation of the software (supporting documents).

2.7 Software Maintenance

The term "software maintenance" is used to describe the software engineering activities that occur following delivery of a software product to the customer.

The primary goal of the software development should be the production of maintainable software systems. The primary attributes that contribute to software maintainability are clarity, modularity, good internal documentation and complete supporting documents.

The same principles applied to the previous phases are applied to the maintenance phase. Enhancement and adaptation of software reinitiates development in the analysis phase, while correction of a software problem may reinitiate the development cycle .in the analysis phase, the design phase, or the implementation phase.

During maintenance, any changes which are implemented must be documented in such a way that the design documents (flow diagrams, etc.), the internal documentation and the user's manual are updated to reflect the changes and new test cases are designed to assess the adequacy of the modification. Updated versions of the software (code and supporting documents) must then be distributed to user sites.

Maintenance activity for a software product usually occurs in response to a change request filed by a user of the product.

A change request may require enhancement, adaptation or error correction. Major enhancements and adaptations should be handled as new projects.

A change request is first reviewed by an analyst. If the problem is related to the use of the program by the user, the user is notified and the case is closed. Otherwise the case should be forwarded to a control board, together with the proposed fix and an estimate of the resources required to satisfy the request.

The change control board reviews, and either approves or denies the requests.

In the case of approval, the software is modified, revalidated and submitted to the change control board for approval. If the control board approves the change, the master code and documents are updated and the modified software is distributed to user sites. If the changes are not approved, the necessary modifications are addressed by the programmers and submitted again.

The composition of the change control board and the formality with which it functions, depend mostly on the nature of the software. In most of the applications with which the mining industry is concerned, the control board will consist only of the individual programmer's immediate supervisor.

In all cases the copies of the request forms are kept in a file to demonstrate the maintenance history of the software. A data base should be prepared to provide information concerning product structure, current revision number, current status and change request history for each products version. The following questions should be answered.

1. How many versions of each product exist?
2. How do the versions differ?
3. Which versions of which products are distributed to which sites?
4. What documents are available for each version of each product?
5. What is the revision history of each component of each version of each product?
6. When will the next revision of a given component of a given version of a product be available?
7. What hardware configuration is required to operate a specific version of the product?
8. Which versions are affected by a specific error?
9. Which errors were corrected in a specific revision?
10. What are the causes of reported product failures?
11. What was the configuration of a given version on a given date?

2.8 Supporting Documents

The following supporting documents are necessary when the software is delivered to the customer:

1. Design Phase Document

The design phase document contains all the design notations such as structure diagrams, flow charts and bubble diagrams which describe the source code. The design phase document must be written according to the specifications described in the previous chapter.

It is obvious that the design phase documents are necessary when the actual source code is delivered to the customer, and the customer maintains the code.

2. Theoretical Description Document

This document presents the theory and the algorithms used in order to arrive at the solutions given by the software.

3. User's Manual

The user's manual is probably the most important document to the end user. It must contain the following information:

1. Introduction

   Product rationale and overview

   Terminology

   Basic features

   Summary of input and output features

   Outline of the manual

2. Getting Started

   Sign-on

   Help mode

3. Data Files

   The units and the formats should be specified clearly and examples should be demonstrated.

4. Modes of Operation

   Commands and options should be specified.

5. Example runs for each option.

6. Advanced features (if any)

CHAPTER 3

SPECIFICATIONS FOR THE SELECTION OF OPERATING SYSTEMS

3.0 INTRODUCTION

An operating system is a program which acts as an interface between the user of a computer and the computer hardware. Its purpose is to provide an environment in which the user may execute programs. The goals of an operating system are, primarily, to make the computer system convenient to use, and secondarily, to use the computer hardware in an efficient manner.

3.1 Standard Operating Systems

Standard operating systems emerged in the 1980's as a result of the standardization efforts in the area of microcomputers. The following significant operating systems were developed:

3.1.1 CP/M

CP/M stands for Control Program for Microcomputers and was developed in 1975. Initially it was intended for an early 8-bit microprocessor, the Intel 8080. CP/M became popular at that time because:

- it was small, requiring only 8K of memory,
- it handled low level input/output tasks, freeing programmers to concentrate on their applications,
- it was portable,
- it was simple and easy to learn.

However this operating system was quite elementary, and although many application programs were written for it, it was substituted by other more sophisticated ones.

3.1.2. MS-DOS

MS-DOS was selected by IBM as the operating system for its IBM PC which was announced in 1981. With IBM's support it soon became the operating system for single user 16-bit personal computers.

At its creation, MS-DOS was defined by the hardware constraints of the IBM PC. The first PC had a memory of 64K and one floppy disk drive. Therefore small size and simplicity were the key goals for MS-DOS. Initially MS-DOS was a little more

than a straightforward imitation of CP/M, reimplemented for the 16-bit Intel 8088.

However, as the microcomputers became more powerful, MS-DOS grew in sophistication.

UNIX provided many ideas for this growth. Figure 9 shows the evolution of MS-DOS.

MS-DOS is a single user operating system. At the present time it is limited to a memory addressability of 640K. This poses a severe restriction to its program support and multitasking capabilities. It must be noted that multitasking is done through Top View and that the total memory of the tasks to be performed cannot exceed the 640K. This makes the utility almost obsolete for engineering applications.

The operating system is easy to learn and many applications are available. It is expected that the next versions of the operating system will be able to address more memory than 640K.

3.1.3 UNIX

The growing popularity of UNIX is closely related to the development of the 16/32 bit and the 32 bit microprocessors.

UNIX however, is an "old" product. It was developed at Bell Laboratories in the late 1960's and early 1970's. Initially UNIX remained primarily a tool for use within the Bell System. Today, versions of UNIX are available to a wider range of

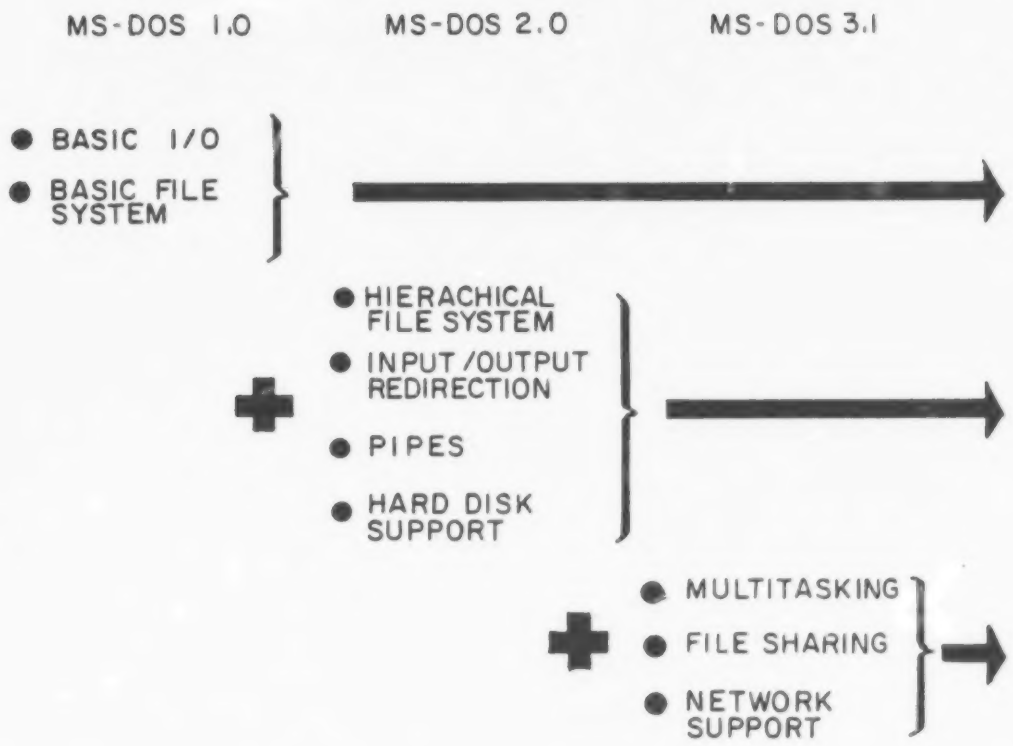


FIGURE 9 : MS-DOS FEATURE EVOLUTION .
AFTER WEINBERG AND GROFF

computers than any other operating system. Unix has been implemented to systems as small as a personal computer and as large as the multi-million-dollar Cray II supercomputer.

Several versions of UNIX were developed since the system was first implemented. Numerous computer science organizations, such as RAND, BBN, the University of Illinois, Harvard, etc., were attracted by UNIX's small size, modularity and design and produced their own versions. The history of UNIX's versions is shown in figure 10. However, AT&T's licensing policies have contributed to the confusion surrounding versions of UNIX. AT&T licenses the UNIX software, but not the UNIX trademark. Therefore, software suppliers and computer manufacturers cannot name their products "UNIX". Instead, each vendor gives a unique name to their product.

The most recent versions offered by AT&T are the UNIX System III (1983) and the UNIX System V (1984). Most of the UNIX systems available today resemble the UNIX III or the UNIX V System.

3.2 The Features of UNIX

UNIX was designed to be a time-sharing system. The file system is a multi-level tree, which allows users to create their own subdirectories.

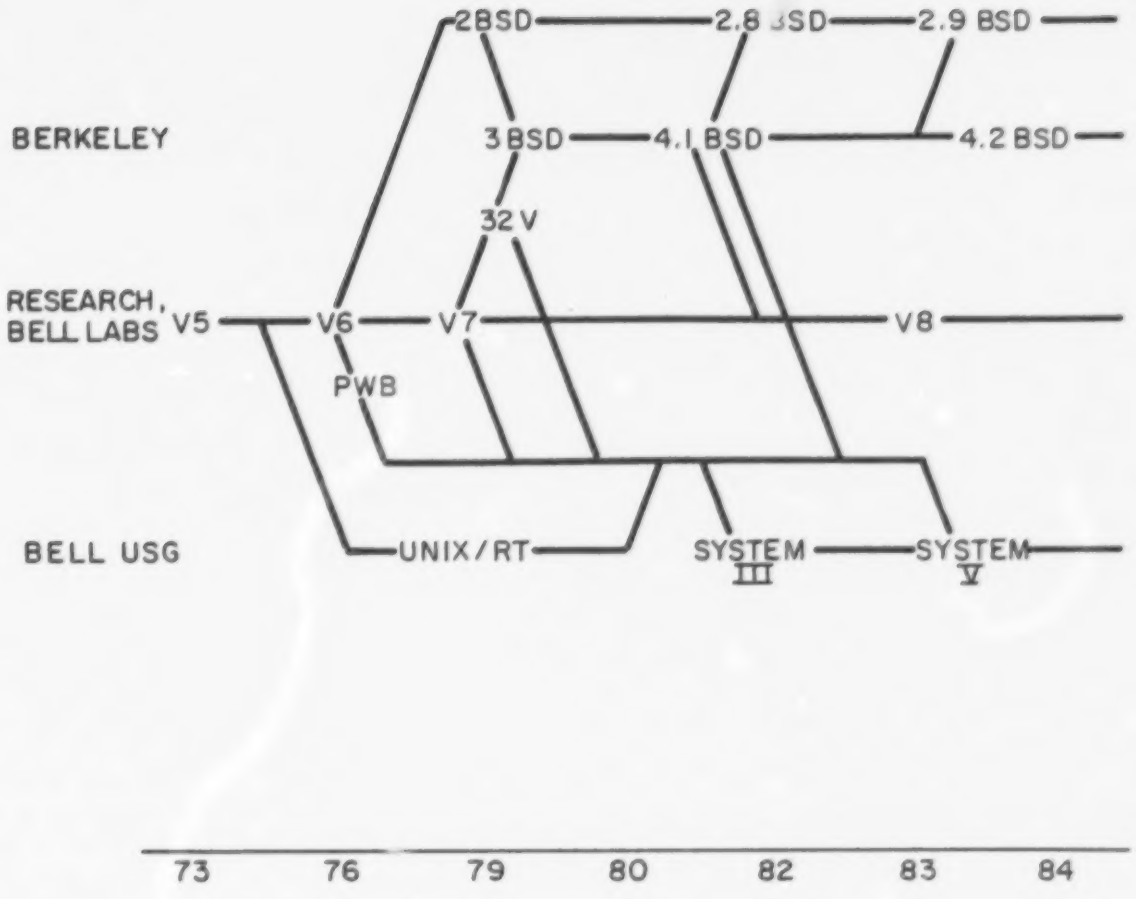


FIGURE 10: HISTORY OF UNIX VERSIONS

Disk files and I/O devices are treated in the same manner. Thus, device dependencies are mostly confined to the device drivers.

UNIX supports multiple processes. CPU scheduling is a simple priority algorithm. Memory management is a variable region algorithm with swapping.

The operating system is mostly written in C. The avoidance of assembly language has greatly simplified the transportability of UNIX from one hardware system to another. The main features of UNIX are the following:

Portability

UNIX is available on different computer systems. Applications software is portable easily. Thus, programs which are written to run on a UNIX system, will run on a variety of different computers.

Multi-user Operations

UNIX is designed to support a group of simultaneous users. The system allows efficient sharing of the processing power and the information storage of a computer system, while offering the security and protection features needed to insulate each user from the activities of other users.

Background Processing

UNIX allows the user to initiate a task and then proceed to other activities while the system continues to work on the original task. Thus, UNIX improves the productivity.

Hierarchical File Systems

The hierarchical file system offers flexibility for grouping information in a way that reflects its natural structure. Data from different users can be used according to the company organization.

Pipes

Pipes can be used to combine several simple programs to perform more complex functions.

Utilities

UNIX includes 200 utilities for functions like sorting data, processing text and searching for information.

Software Development Tools

UNIX provides an extensive array of software tools that support all phases of the development process, from program editing to debugging. UNIX is used extensively to develop systems of applications software.

Maturity

The software is mature and free of errors, offering a high level of reliability.

3.3 UNIX Versus MS-DOS

The comparison between MS-DOS and UNIX is changing with time. MS-DOS has slowly evolved into a sophisticated operating system incorporating many of the major features of UNIX. However, there are important differences between them. The

differences between the two operating systems are shown in table 9.

MS-DOS is a single-user operating system. It provides the software foundation for a PC network, which is accomplished by linking personal computers. This is shown in figure 11.

UNIX is a true multiuser system. It is designed for sharing computer power and common data among the users in a work group. A single computer is shared among multiple users according to the UNIX approach. This is shown in figure 12. The disadvantage of the multiuser approach is the performance degradation the system suffers, as more and more users are added to it.

PC networks on the other hand are not a perfect solution for sharing data. Files can be easily exchanged but coordinating true concurrent access to a single copy of shared data is a difficult task.

Moreover, although DOS has approached UNIX, the level of sophistication in the utilities it offers is elementary when compared to UNIX. Many of UNIX utilities implement functions that are not available in DOS.

An example of this difference is the UNIX utility 'grep', which searches text files for occurrences of a particular pattern. MS-DOS has the utility 'FIND' which is similar. However, 'FIND' allows matching on only a literal string of characters, whereas 'grep' supports much more flexible pattern matching. 'grep' can locate lines in a document that

MS-DOS	XENIX
- SINGE USER	- MULTIUSER
- MEMORY-BASED OPERATION	- DISK-BASED OPERATION
- PERSONAL DATA	- SHARED ACCESS TO COMMON DATA
- SCREEN-INTENSIVE APPLICATIONS	- DISK-INTENSIVE APPLICATIONS
- MULTITASKING FOR RELATED TASKS (TOP VIEW)	- MULTIPROGRAMMING FOR RELATED AND UNRELATED TASKS
- FRIENDLY FOR END-USERS	- FRIENDLY FOR PROGRAMMERS

TABLE 9: A COMPARISON OF MS-DOS AND XENIX.

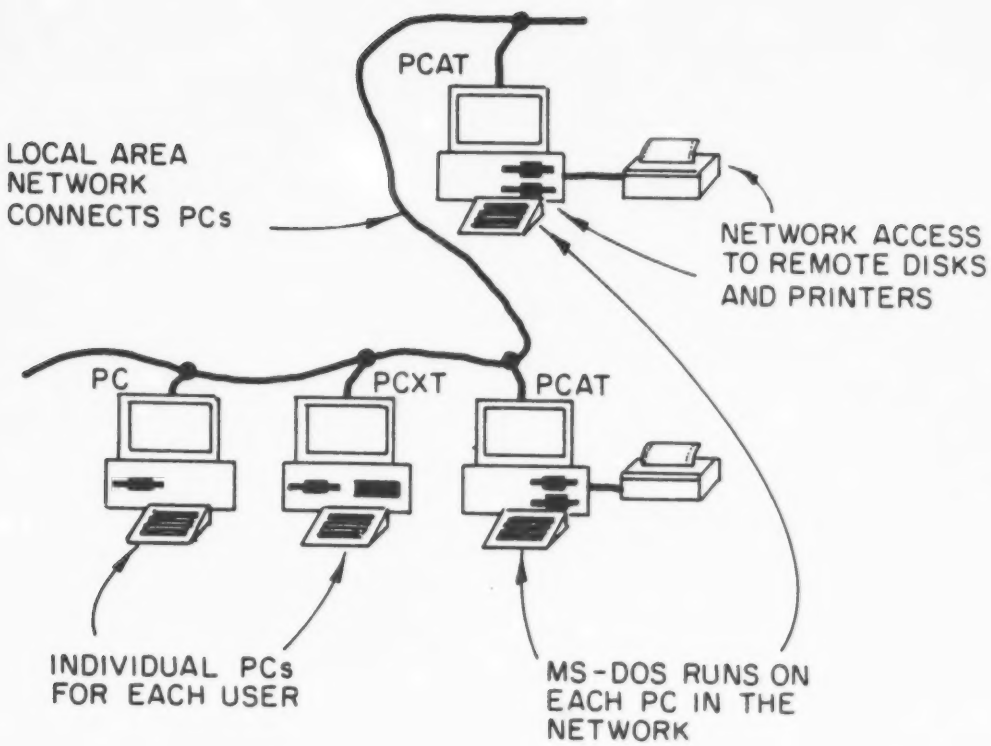


FIGURE 11 : PC NETWORKS - THE MS-DOS APPROACH.

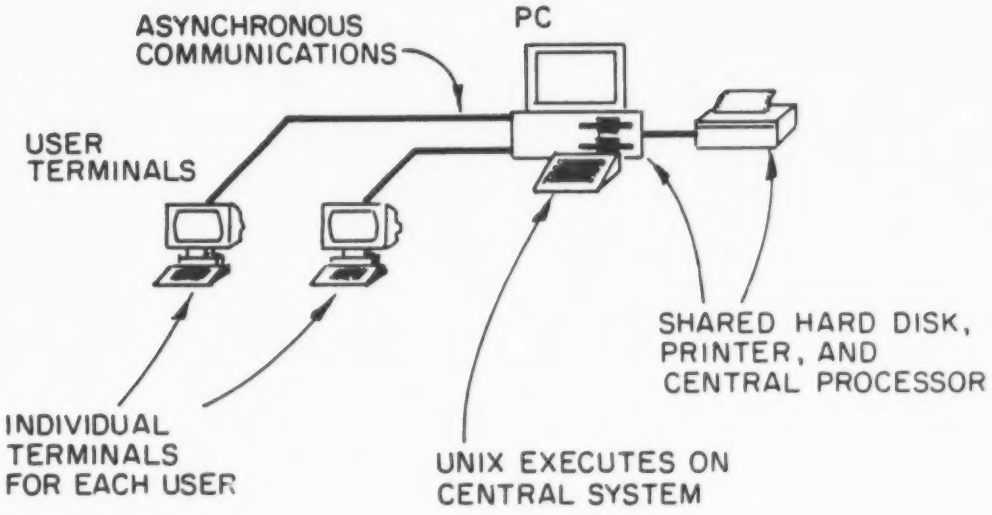


FIGURE 12: MULTIUSER SYSTEMS—THE XENIX APPROACH.

contain two words, even if they are separated by one or more intermediate words.

Recently UNIX has incorporated a system which allow the user to run MS-DOS applications under UNIX.

Moreover software has been developed for networking UNIX-based microcomputers.

Software developers have introduced systems which take advantage of UNIX's structure. One of the important aspects of UNIX is the concept of file/device transparency. The same set of utilities can be used on I/O devices, directories, ordinary files and even main memory. All devices appear as ordinary files and behave as such.

It is now possible that applications run in the network, as if the network were a single machine. This is accomplished by extending UNIX's tree like hierarchical file system with a new super root directory. The user can execute a program using a processor other than the one to which he is currently connected. This capability is very important in a large network with a variety of computers. The most powerful processors can be assigned time consuming jobs such as running finite element applications, while the less powerful processors can run ordinary applications.

CHAPTER 4

THE SELECTION OF A SINGLE PROGRAMMING LANGUAGE CAPABLE OF CODING ALL MINING RELATED PROGRAMS

4.0 INTRODUCTION

A programming language permits people to communicate with computers through the use of symbols, characters and rules. There are many types of programming languages; special purpose languages and high-level general purpose languages.

Machine language, or binary, is the only language the computer's central processing unit directly understands. Each instruction in machine language is composed of at least two parts: an operation code, or op-code, that tells the computer which function to perform, and an operand, that tells the computer where to find the next piece of information required.

In order to simplify and speed up the process of programming, assembly languages were developed. The assembly languages involved the use of letter symbols for the numeric machine language instruction codes. In this sense the programmer did not need to use the binary code, or series of 0's and 1's, to represent all instructions. This is a significant advantage for the programmer. Further advancements included machine language coding so that symbols could be used to locate positions in memory. Machine language is still used by the computer. The

assembly language is merely used to convert symbols which are familiar to people, to the binary code necessary to the computer. A program or code of instructions is referred to as the source code. A program, or code of instructions that have been translated to machine language, or binary, is referred to as object code.

Assembly languages have many advantages over machine language. Assembly language involves using more familiar symbols. Thus it saves time, reduces errors, allows easier modification, and is more friendly in general for all people dealing with the code. There are drawbacks however, and it is for these reasons that 3rd generation higher-level languages have been developed. Coding in assembly language is still time consuming and as more powerful machines were developed, more familiar symbols were developed.

The early assembly language programs produced only object code instruction (machine language) for each source code instruction. However, as the assembly languages matured, it was possible that a single macro instruction could cause several or many machine language instructions to take place. As an example, the READ or WRITE commands are composed of many machine language instructions.

The further development of these macro instructions led to the development of the third generation of computer languages which are known as high-level languages. The high-level languages have the benefit of being able to be used with many

different makes of computers, with little if no modification. The high-level languages have several other major advantages over the assembly languages as follows:

- they require less time to write
- they are easier to maintain
- they allow better documentation
- they are easier to learn than the assembly languages
- a programmer skilled in writing programs in the high-level language is not restricted to using a single type of machine.

Obviously, source code written in the higher-level languages must be converted into machine understandable object code before the program can be executed. To do this, the higher-level languages use a compiler. The compiler, in the translation from source code to object code, generally makes many object code instructions from each source code instruction. It is evident that the compiler must be run before the program can be executed.

Some of the high-level languages have elected to use a different approach to the use of a compiler. In these languages an interpreter is used in place of a compiler. This technique has proven valuable for some personal computers. As a program is under execution, each source code instruction is translated into the corresponding object code instructions. In this fashion the entire source code is not translated and stored in memory as an object code file; only those object code instructions being performed are stored in memory at one particular time. The

largest single advantage of the interpreter type language is the smaller amount of memory required during execution. However, if memory is available to store the object code and the program is to be re-run, the interpreter type language should not be used, because each individual source code instruction must be retranslated into the object code instructions. This has the inherent disadvantage of being much slower and using considerably more computer time to perform the same function.

Having described the evolution of the high-level languages, the various languages currently in use must be examined. To describe and compare each language individually is not in the scope of the present report. Therefore, a small description of each program language that must be considered for engineering and scientific programming in the Mining Industry will be made, followed by a more detailed comparison of the languages selected on the basis of such factors as program structure, layout and declarations. The languages selected for comparison purposes are: BASIC, C, FORTRAN, PASCAL, and PL/1. Many other languages exist and have been examined.

An examination of COBOL proved it to be an excellent language for business purposes, however, it was not particularly well suited to engineering and scientific applications. COBOL is very capable when manipulation of alphabetic characters is required. Its major limitation, and the major reason for not considering COBOL, was its unsuitability for complex mathematical manipulations.

APL was not compared to the other languages mainly due to its data structuring restrictions. Array declarations and character declarations are either not needed or not required and therefore input/output operations are restricted to simple input and output of numbers, character strings and arrays.

4.1 Language Review

BASIC

The name BASIC stands for "Beginners All-Purpose Scientific Instruction Code". BASIC is an interactive computer language that enjoys great acceptance by beginning programmers. Data input/output is easy and the format is often provided automatically. BASIC is one of the most popular high-level languages used in personal computers. It is available as both an interpretive version and as a compiler version. An American National Standards Institute (ANSI) standard was established for BASIC in 1978. However these standards are so minimal that a universal BASIC does not exist.

C

C was developed at Bell Laboratories and was used to write the UNIX operating system. It has become a favorite of systems programmers and others who develop software packages for

small computer systems. It is also used to create graphics and special effects in films. Coding in C often approaches machine language in density and efficiency. C has been referred to as a high-level assembly language. C does offer many high-level language features and has gained a significant number of supporters since its introduction in the mid 1970's. For applications that require more direct "contact" with the computer and its memory, C has been by far the most popular language selection.

American National Standards are being developed for C at the current time. Hence, one of the major drawbacks for the language, is the lack of a standard for software developers use.

FORTRAN

FORTRAN has long been the standard language used for engineering and scientific programming. The origin of FORTRAN can be traced back to 1954 to FORmula TRANslator when an IBM sponsored committee began work on a scientific mathematical language. FORTRAN has been praised for the ease with which it can express mathematical equations. FORTRAN coding makes use of a compiler rather than an interpreter. Standardization of the language began in 1962 and to date two standards have been developed. FORTRAN was the first standardized language. The current FORTRAN standards were developed in 1978. FORTRAN has been well accepted by the engineering and scientific community.

For this reason very large libraries of engineering and scientific programs exist.

PASCAL

PASCAL is an offspring of the ALGOL language developed in the 1950's. PASCAL was the first major language to be created after the concepts of structured programming became widely accepted. PASCAL can be used for both engineering and scientific applications and business applications. PASCAL is block structured. All variables are identified at the beginning of the program. The three basic logic structures are accommodated by the statements IF...THEN...ELSE and WHILE...DO loop statements. PASCAL is one of the major languages running on mini and microcomputers. An American National Standard was developed for PASCAL in the early 1980's.

PL/1

In the early 1960's, IBM, and a committee of the IBM 360 users, recognized that programming the languages were either developed for engineering and scientific programs or for business data processing applications. The PL/1 language was designed with this in mind and was implemented in the mid 1960's to solve all types of both scientific and business problems. PL/1 (Programming Language One) was designed to include some of the

features of FORTRAN and make use of COBOL type file processing techniques. The American National Standards Institute established a standard for PL/1 in 1976. Because PL/1 is composed of features from both FORTRAN and COBOL it is a flexible and sophisticated language. Modular procedure blocks and other features of the language support the use of the structured programming concepts.

The above five languages have been examined in eighteen different areas. The examination involves drawing comparisons between each language and isolating any similarities or differences that exist. Table 10 provides a comparison of the languages which were examined.

A comparison of the programming structure, dictated by each of the programming languages, would indicate that C, FORTRAN and PASCAL provide an ordered and logical programing structure. C is preferential to the other languages by virtue of its programs being composed of many small functions. Those functions may be compiled separately, saving enormous amounts of compilation time. Many programs may call the same functions, thus simplifying the use and access of functions usable in many programs.

Statement layout is quite similar among the C, FORTRAN, PASCAL and PL/1 languages. Statements may be continued from line to line if needed and programmer freedom exists. BASIC, on the

	BASIC	C	FORTRAN	PASCAL	PL/1
PROGRAM STRUCTURE	Continuous block of statements, functions and subroutines; part of program block. LACKS STRUCTURE	Composed of numerous small functions. A program may reside on one or more source files, the source files may be compiled separately and loaded together along with previously compiled functions from libraries. ORDERED AND LOGICAL STRUCTURE TO FOLLOW	The FORTRAN program consists of main programs and a number of separately compiled subprograms. ORDERED AND LOGICAL FLOW	PASCAL programs are composed of a data definition section, executable block definitions and an executable main program block. The main program block is the first block to be executed when the program is run.	A PL/1 program consists of one or more separately compiled external procedure blocks. HIGHLY STRUCTURED
STATEMENT LAYOUT	Only one line per statement is allowed however more than one statement can appear on a single line.	Written in free form with any number of spaces and/or new lines used to separate the words and symbols. Each statement must be separated from the next statement by a semi-colon.	Columns 7-72 are used for statements. May continue statement on another line by placing a character in column 6.	Each statement must be specifically separated from the next by a semicolon, except the statement just before the end statement.	Statements are written in free from with any number of spaces and/or new lines used to separate words and symbols.
DATA TYPES	Numeric and string characters are the only data types. Some more powerful versions are available in which single and double precision are available.	There are a few basic data types Char: a single byte holding a single character Int: an integer Float: single precision floating point Double: double precision floating point.	In FORTRAN the data types are Integer 124, 15000 Real 1.26, -1027 Double Precision 1.276 D-14 Complex 5.21, 3.1 Logical 'Jack' Character 'A', 'g'	In PASCAL the four data types are: Integer 124, 15000 Real 1.26, -1027 Boolean true, false Character 'A', 'g'	In PL/1 the data types are: Arithmetic Character string Bit string Entry File

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES. (continued)

	BASIC	C	FORTRAN	PASCAL	PL/1
IDENTI-FIERS	Identifiers are names denoting variables and constants.	Identifiers are a sequence of letters and digits. Traditional C practice is to use lower case variables for variable names and upper case for symbolic constants.	FORTRAN identifiers are names denoting constants, variables, functions or subroutines.	PASCAL indentifiers are names denoting constants, variables,procedures.	PL/1 identifiers are names denoting constants, problem variables and program control variables.
DECLAR-ATIONS	Declarations are names denoting variables and constants.	All variables must be declared before use. This ensures consistency throughout the programing.	Explicit declaration of variables must be done with the INTEGER, REAL, DOUBLE, PRECISION, COMPLEX, LOGICAL and EXTERNAL statements.	Declaration is done in the data definition section for all identifiers.	Declaration of variables is through the DECLARE statement.
STRUCT-URED TYPE ARRAY	Arrays can be either one or two dimensional. Rules for naming numeric and string arrays are the same as those for identifiers.	Arrays can be multi-dimensional. C provides for rectangular multi-dimensional arrays, although in practice they tend to be much less used than arrays of pointers.	Arrays can be multi-dimensional. Rules for naming are the same as those for identifiers.	Multi-dimensional arrays are provided for. Rules for naming are the same as those for identifiers.	Multi-dimensional arrays are provided for with the same naming rules as those for identifiers.

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES. (continued)

	BASIC	C	FORTRAN	PASCAL	PL/1
ARRAY DECLARATIONS	Arrays that need more than 10 storage locations must be declared in a dimension (DIM) statement.	Array component type and index type are specified in the data definition section.	Array sizes are generally declared with the dimension section.	Array component type and index type are specified in the data definition section.	sizes of arrays are declared with a DECLARE statement. The attributes of each array may follow each size declaration.
OPERATORS	The arithmetic operators are: Exponent * / + - =,< >, <, < =, >, > = and, or not.	The arithmetic operators are: + - * / >, > =, <, < = = = equal ! = not equal & & and | | or many other operators are available flat often control on a bitwise basis.	The arithmetic operators are: * * Exponent * / + - .EQ. .NE. .LT. .LE. .GT. .GE. .NOT. .AND. .OR.	The arithmetic operators are: NOT * / DIV MOD AND + - OR =,< >,<,< =,>,> =	The arithmetic operators are: * * * / + - || Concatenation NE.,LT.,LE.,NL., GT.,GE.,NG., > NOT & AND | OR * POINTER

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES. (continued)

	BASIC	C	FORTRAN	PASCAL	PL/1
EXPRESS-IONS	Numerical quantities are represented by arithmetic expressions, while Boolean quantities are represented by logical expressions.	Numerical expressions are represented by arithmetic expressions while Boolean quantities are represented by logical expressions.	Numerical expressions are represented by arithmetic expressions while Boolean quantities are represented by logical or Boolean expressions.	Expressions are constructed through the use of: 1) the grammar of expression which is the familiar index notation. 2) the type of data. The type data affects the kinds of operators allowed in the expressions.	PL/1 expressions may be element (scaler) expressions which refer to single items of data or array and structure expressions which refer to arrays and structures.
ASSIGN-MENT	Assignment is achieved with the LET statement or with a combination of READ and DATA statements.	Assignment is achieved with the assignment statement.	Assignment is achieved with assignment statements and DATA statements.	Assignment is achieved with the assignment statements.	Assignment is achieved with the assignment statement.
CONDI-TIONAL & UNCONDI-TIONAL BRANCHING	Unconditional branching is achieved through the use of GO TO and GO SUB statements. Conditional branching makes use of the IF THEN ELSE construct.	Unconditional branching is done using the "go to "label"" statement. Conditional branching is achieved using IF ELSE, ELSE IF or the switch constuct. The switch statement is a special multi-way decision maker.	Unconditional branching is achieved using GO TO statement. Conditional branching is achieved using combinations of the arithmetic IF statement.	Unconditional branching is achieved with the GO TO statement. Conditional branching is done through the use of IF-THEN-ELSE and case constructs.	Unconditional branching is done with the GO TO statement. Conditional branching is achieved using the IF THEN and IF THEN ELSE statements.

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES. (continued)

	BASIC	C	FORTRAN	PASCAL	PL/1
LOOPING/ ITERATION	Looping or iteration is achieved using the FOR NEXT statement.	Looping or iteration is done three ways in C, using the FOR, WHILE, and DO constructs.	Looping or iteration is done using the DO statement at the start of the loop and a numbered statement at the end.	In PASCAL there are three forms of the loop statement the FOR, WHILE and REPEAT constructs.	In PL/1 the DO statement is used at the beginning of the loop and the END statement at the finish.
FUNCTION	Single line and multiline functions are available. All function declarations begin with a DEF statement and with a FNEND statement.	A function in C is similar to a subroutine of function in FORTRAN. A function provides a convenient way to encapsulate some computation in a black box.	A function whose value may be computed in a single arithmetic or logical expression may be defined as a statement function, local to a particular subprogram. Functions with more than one statement are compiled as separate subprograms.	Function declarations begin with the word FUNCTION. The type of the function and the parameters are specified in the heading of the declaration.	A procedure block is headed by a PROCEDURE statement and terminated by an AND statement.
SUB-ROUTINE	A subroutine is called using the GOSUB statement. The subroutine ends with the RETURN statement.	In C the function is used in the same manner as the subroutine.	In FORTRAN there are two types of subroutines. The function subroutine returns one value only to the main program where as the subroutine subprogram may return more than one value.	A subroutine has more than one statement and is compiled as a separate subprogram. It begins with a SUBROUTINE statement and ends with an END statement.	A procedure is a self-combined program segment starting with the word PROCEDURE. Procedures must be defined before their use unless they are external or defined in a FORWARD statement.

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES. (continued)

	BASIC	C	FORTRAN	PASCAL	PL/1
INTRINSIC/ LIBRARY FUNCTIONS	ABS (X) Absolute value SIN (X) Trigonometric function COS (X) Trigonometric function TAN (X) Trigonometric function ATN (X) Trigonometric function INT (X) Integer part EXP (X) Exponential function e LOG (X) Natrol logarithm LGT (X) Common logarithm SQR (X) Square root SGN (X) Sign of X RND (X) Generates random number	The library of C functions is very extensive and includes all those functions listed for FORTRAN and more. Library functions may be accessed in the same manner as user defined functions.	ABS (X) SIN (X) COS (X) TAN (X) ATAN (X) EXP (X) Exponential function e ALOG (X) Calculates natural logs ALOGIO (X) Calculates common logarithm SQRT (X)	ABS (X) SIN (X) COS (X) ARCTAN (X) EXP (X) LN (X) SQRT (X) SQR (X) ODD (X) EOLN (X) EOF (X) TRUNC (X)	PL/1 has a very large library of built in functions composed of those of other languages.
INPUT/ OUTPUT	I/O is achieved using the INPUT and PRINT statements. Data is entered in free form with commas as separators.	C has no input or output facilities. READ WRITE command must be called by specifying explicity called functions. A list of all the standard functions may be obtained from any text on the language C.	I/O is achieved using READ, WRITE and FORMAT statements.	I/O statements are as follows: READ, READLN, WRITE, WRITELN GET and PUT	I/O statements are as follows: READ, WRITE, GET PUT, LIST,DATA, EDIT.

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES. (continued)

	BASIC	C	FORTRAN	PASCAL	PL/1
PROGRAM HALTS	The STOP statement halts the program execution, however, the variable values remain active.	C does not have a halt type command.	The STOP statement halts execution of the program. The PAUSE command temporarily halts execution.	PASCAL does not have a halt type command.	The STOP command halts execution of the program whereas the DISPLAY, DELAY, HALT, WAIT, statements temporarily halt execution.
DOCUMENTATION	Documentation may be provided using the REM (remark) statement.	Documentation my be provided using the 1* *1 statement. Comments can appear anywhere.	Documentation may be provided. A C in column 1 indicates that particular line is a comment line.	Documentation may be provided using the (* *) statement.	Documentation may be provided using the 1* *1.

TABLE 10: COMPARISON OF FIVE PROGRAMMING LANGUAGES.

other hand, is limited to requiring each statement to occupy only one line of coding.

The elementary data types available for the five programming languages vary considerably. Both the programming languages C and FORTRAN provide for the most distinctly defined data types. Double, single precision and floating point numbers may be specified as well as character data types.

The identifiers used in all five of the selected programming languages are very similar.

The declaration of all variables prior to use is a necessity for consistent and efficent programming. Declaration of variables prior to first use enforces consistency of usage of a particular variable throughout the program. Not declaring variables allows inconsistent usage of variables, and increases the probability of errors. The four languages C, FORTRAN, PASCAL and PL/1 enforce some type of character declaration. C enforces the declaration of variables to a much greater extent than the other languages.

Array types for the five languages reviewed are either single or multidimensional. Declaration of arrays is made either through a dimension statement or in the data definition section.

The basic arithmetic operators are similar for the five major languages considered. C provides for a much greater library of arithmetic operators, that allows control on a bitwise basis. Thus, it allows control which is much closer to the actual machine operation if required.

C, FORTRAN, PASCAL and PL/1 all achieve assignments through the use of an assignment statement. BASIC on the other hand uses a LET or READ and DATA statements to accomplish assignment. Recent versions of BASIC have made the LET statement optional.

For all five languages considered, unconditional branching is achieved through the use of the GO TO statement. Conditional branching is provided for in all five languages through various combinations of the IF THEN ELSE statements. C provides one additional form for conditional branching in the form of a 'switch' statement that may be used for a more specific comparison if needed.

Looping or iteration is achieved in much the same manner for the five different languages. However, both C and PASCAL offer a greater variety of looping or iteration forms. C and PASCAL offer three forms of iteration, while the other languages offer one. This greater choice for the C and PASCAL languages allows expression to be evaluated in a more precise and efficient format.

Subroutines are provided for in all five languages evaluated. C uses the function statement in the same manner as subroutines are typically used for. The function should be composed of a single idea or logic flow. FORTRAN has two types of subroutines: the function, which only delivers one particular value to the main program and the subroutine, which may return multiple valves.

All five languages have intrinsic or library functions which may be called in the same manner as the defined functions. These library functions are quite similar although C has a very extensive list.

BASIC, FORTRAN, PASCAL and PL/1 all have similar input/output facilities built into the programming languages. Input is generally achieved using the READ statement, and OUTPUT using the WRITE statement. C, however, has no standard input/output facilities. The input/output facilities are functions that must be explicitly called. This is not a restriction of the C language; instead, this often increases the portability of C programming, as will be detailed latter in this report.

Documentation may be provided for all five different programming languages.

4.2 Discussion of Language Review

A review of the preceeding comparison of the five languages BASIC, C, FORTRAN, PASCAL and PL/1 would indicate that the languages are similar in many respects. However, there are many critical differences that exist between each of the languages.

The BASIC programming language was designed with the first time programmer in mind. The language was developed to be powerful enough to be used for small and medium scale problem

solving. However, for large scale work the language is deficient in many respects.

PL/1 was developed as a multipurpose programming language, based upon FORTRAN, that was easy for the beginning programmer to use. With this in mind PL/1 was constructed so that the individual programmer could program at any level of experience. The language has many automatic and default conditions to assist the beginner, while the experienced programmer may detail each and every step of a complicated program. Unfortunately, the default condition concept is more confusing than helpful to the programmer, as conditions may not be the ones expected. PL/1 suffers from being too broad of a programming language, due to its multipurpose, multifault design philosophy. This philosophy has actually made the language more difficult to learn in its entirety, and difficult to read and write.

The rest of the programming languages examined are discussed in more detail in the following.

When selecting a computer language, much depends upon the subject areas of concern. While traditionally engineering and scientific programming in the mining industry has been of the numerical type (simulation, and finite element analysis), the scope is much broader today, and will continue to grow. Engineering and scientific programming today, involves data processing and spread sheet type programming, robotics and

process control type programming, as well as the more traditional numerical processing.

For the more traditional numerical type programming FORTRAN and its facilities proved to suffice. FORTRAN was the first high level computer language developed. Many engineers and scientists became familiar with the language before other options were developed. Many programs exist in the FORTRAN language today. For these reasons a resistance to change has developed, and, FORTRAN is often the only language considered when new programming applications are perceived.

Because of its mathematical nature, FORTRAN limits its expressions of data structures to simple variables and arrays, the latter being the only real structure that allows a programmer to organize complex data. Integers, single and double precision, floating point numbers, complex numbers and Boolean (logic) expressions are all orientated towards an easy way for the engineer or scientist to express numerical data to solve problems.

However, for handling characters in FORTRAN, the programmer inherits many limitations. FORTRAN has not been optimized to handle input or output as in the case of languages such as COBOL and PASCAL. This is a major drawback of FORTRAN, because describing records or files is much more difficult. Input/output facilities built into a language such as FORTRAN, often limit portability by introducing machine dependencies to the computer coding. Machine dependencies occur due to a piece

of code that is specific to one particular piece of hardware. This limitation typically occurs in input or output routines.

FORTRAN also suffers in certain respects from having been adapted as technology has developed. From the time FORTRAN was conceived, many developments have taken place. New versions of the FORTRAN language have been adopted over time. These new versions however, must be somewhat compatible with previous versions. For this reason the updated versions do not fully take advantage of hardware and other language developments.

C is a general purpose programming language originally designed for and implemented on the UNIX operating system. The C language has been closely associated with the UNIX operating system, however, it is not tied to any operating system or machine. C has often been referred to as a system programming language, as its abilities extend to writing operating systems. C has also been used to write numerical text processing and database type programs. Its absence of restrictions and its generality make it more convenient and effective for more tasks than other powerful languages.

C has been referred to as a "low level" language. This does not mean it is another assembly type language; rather operations may be performed closer to the actual hardware of the computer than with many other languages. C deals with the same sort of objects that most languages do (characters and numbers); however, the ability to operate the hardware on an address basis

is also present. For certain types of programming, efficiency and complete control of the processor is required. These programming types might include robotics and process control, subjects typically programmed in assembly languages. Because C allows bitwise control, many, if not all of these applications, may be done in the C language.

C is a language developed around the function concept. C provides no operations to deal directly with composite objects such as character strings, sets, lists, or arrays considered as a whole. Instead all of those facilities are provided for by functions. By using so called functions in place of other high-level languages, integrated design portability among hardware is simplified.

C is a small language, it can be described in a small space and learned relatively quickly. Due to its small size, compilers may be prepared for new machines quickly. This tends to provide for a high-degree of mobility.

The C language is the most versatile and portable of the languages examined in this study. The ability to design and implement such a wide variety of programming needs is most effectively done using C.

PASCAL has been hailed as the most important contribution to programming since the onset of structured programming. PASCAL was developed to make available a language

to teach programming as a systematic discipline based on fundamental concepts. PASCAL is fast becoming a major teaching language in institutions in both North America and Europe.

The concept of using a highly structured programming language as a teaching aid cannot be faulted. By installing structured programming from the onset, programmers themselves may gain an appreciation of the benefits of structured programming. However, though a high degree of structure is invaluable as a teaching aid, as programmers become more experienced, they tend to desire certain freedoms in their programming style. The forced structure thus limits the freedoms, and in many cases decreases the effectiveness and efficiency of the experienced programmer.

The American National Standards Institute (ANSI) has standardized PASCAL. However, the standard developed has not been widely accepted. Many different versions and extentions have been created that have not obtained the sanction of either ANSI or any other official standards organization. The many different versions existing have limited the use of PASCAL as a software development language.

For writing computer code that requires low-level control of the machine, PASCAL may be used in conjunction with assembly language. The section of the code that requires machine control can be coded in assembly language, and the remainder of the code written using PASCAL. This technique suffers however, due to machine dependencies introduced by the assembly language

coding. These machine dependencies limit the portability of the coding from machine to machine. This is a major drawback of the PASCAL language. C on the other hand does not require the use of two languages to perform low-level operations; rather the language itself incorporates many of the low-level features of assembly languages.

4.3 Language Selection

From the general review of many computer languages, and the detailed examination of five major computer languages, C was selected as the most desirable language for the multipurpose programming the mining industry requires. By the 1990's the scope of engineering and scientific programming will have broadened to include robotics control, expert systems coding, machine controlled rock mechanics surveillance, and, many other areas including the traditional "number crunching" type programming. For this broad range of programming, nature the FORTRAN language will impose severe restrictions and limit programmer efficiency.

C, however, does not restrict the nature of the programming effort and is a general purpose language. It has been used by systems programmers for coding operating systems and by data base programmers for coding data base programs. It can also be used for a host of other applications such as text processing, machine control and numerical processing.

It must be recognized, however, that many high calibre programs exist in the FORTRAN language. It is not a recommendation of this study to convert these programs to the C language. Rather, only new programs should be coded in the C language. However, the programs existing in FORTRAN, which require major updating and revisions (greater than 50% of the coding rewritten) should be converted to the C language. In this manner, the C language will be phased in gradually, over a period of several years, and eventually most, if not all coding, will be written in C.

This introduction of the C language will require both attention and perseverance by mine and government management personnel. Programmers will require C language workshops to become conversant with this language. It must be realized that there will be a tendency for programmers to program in FORTRAN, stemming from a resistance to change and a better understanding of that language.

In the medium term the mining industry will benefit immensely from the usage of the C language. The selection of this language allows most mining related programs to be coded in a single language, and increases the portability and uses of each individual program.

CHAPTER 5

5.0 COMPUTER GRAPHICS

5.1 Introduction

The visual presentation of data on a computer has evolved from bland representations in textual form to vividly descriptive forms which approach lifelike images. Computer graphics has afforded us a more appealing means of displaying, interpreting and communicating numbers which have traditionally been used as the only means by which information could be portrayed by a computer. Computers can be programmed to convert vast amounts of data into forms which depict their meaning in more natural ways. These forms can then be reproduced and manipulated at high speeds. Today computer graphics represents an exciting and fast growing subject area within the field of computer science.

5.2 The Scope of Computer Graphics

The scope of computer graphics extends from the generation of histograms, pie charts, graphs and bar charts through to the visual depiction of industrial plant flowsheets to

the replacement of the functions of pencils, paper, and rulers in system and structure design and analysis.

Three areas of computer graphics have been identified:

- Generative computer graphics
- Picture analysis
- Picture processing.

These areas are dependent on the manner in which data are converted and implanted in a computer, how this data is visually represented, and the types of objects which this data represents.

- Generative Computer Graphics

Pictorial images are constructed from descriptions which are given by computer programs and data. The basic information used may be input by the user, calculated or obtained through commands issued by a user of a graphical workstation. This form of computer graphics includes the construction of images from predefined menu cards as in the case of most CAD systems.

- Picture Analysis

Picture analysis deals with the extraction of objects, shapes and elements with special characteristics from a picture in an unstructured form. Usually the data representing the pictorial image is given to the graphics system by digitizing a photographic picture. This type of computer graphics is related

to what is more popularly known as image analysis. It includes the identification and characterization of objects by their physical properties. Examples of picture analysis includes the identification of different mineral species from images of rock samples obtained from optical microscopes.

- Picture Processing

Picture processing refers to aspects dealing with changing the visual representation of a picture so the human interpretation of the information included in the picture is made better. Examples of these techniques include the enhancement of aerial photographs through the colouring of different landforms and vegetation.

In the Mining industry our requirement for graphics is often a hybrid of the forms identified above. These elements however provide us with a means of classifying our graphics needs so that tools and techniques needed for a productive graphics environment may be specified.

5.3 Computer Graphics Applied to the Mining Industry

Computer graphics forms an integral part of software designed for all stages of mine exploration, development and operation. Computer graphics has proven itself indispensable in

each of the following activities:

- Exploration
- Mine development
- Mine production planning
- Mineral processing
- Management support systems

Each of these activities imposes new demands on computer graphics technology. 2-D generative graphics in the form of maps, graphs and charts plays an important but minor part in the graphics requirements of mining software. Special needs include picture processing and analysis in remote sensing and digital image processing, 3-D object transformation for modelling, sophisticated line drawing and model representations.

The following identifies the special graphics needs of each of the above-mentioned activities which forms an integral part of mine development and operation.

Exploration

Pictorial representation of data from exploration activities requires that graphics software is capable of mapping contours, presenting sections, colour enhancement, image analysis, manipulation and display of 3-dimensional representations.

Mine Development

Primary needs in this area include graphical representation of models of ore bodies in 2-D and 3-D and the ability to perform transformations on these images. In addition to this, much of the requirements of exploration must be met. Mine development focuses on generative graphics. Representative data is retrieved and displayed in many forms, including gridded and contoured topographic maps of open-pit mines, and 3-D block model representations and design layouts. Isometric and orthogonal viewing of models with user control of viewpoint, orientation, windowing and zooming like simulations, require frequent redrawing of these representations as models are updated; this requires improved hardware performance.

Mine Production Planning

In this stage, generative graphics is again found to be of primary application with data entry being made via a keyboard or through digitizing mine plans. Sophistication in computer graphics for this activity is to the most part the same as mine development.

Mineral Processing

Graphics requirements for mineral processing are not as demanding on graphics technology as some of the other mining activities reported above. Graphical representations in mineral processing are usually in the form of images of plant flowsheets and the display of data for process control. New subject areas such as the application of image analysis to mineral dressing requires the use of picture processing techniques.

Management Support Systems

Management support systems include graphical representation of financial, maintenance, metallurigcal and other production data. This form of graphics support usually takes the form of graphs.

5.4 Issues Concerning the Integration of Graphics in Software in the Ontario Mining Industry

For graphics-based computer programs to achieve effective utilization the following issues must be examined:

- The portability of the graphics capability of the software.

Today we are faced with graphics devices from many different suppliers, each with its own communication protocols

and hardware characteristics. Ideally graphics based software should exhibit device-independence. With this feature programs may be ported very easily between different graphics hardware environments.

- Graphics software should remain applicable as technology changes

Again this requires device-independence. In the past graphics software coded for vector display systems slowly became obsolete as raster systems forged to prominence. A modular approach to the addition of graphics capability will improve software maintainability as technology changes.

- Is database adequate for graphical derivations?

Assuming the existence of a central relational database. Is data pertinent to the generation of graphical images present, or can they be derived?.

- Are these programs easy to develop and are they easy to use?

Specifications for the addition of graphics capability to computer software should not complicate the process of software development, nor should the completed programs be difficult to use. Even though the addition of graphics capability to software stimulates thoughts of user friendliness, this does not always become reality. Graphics based software should be easy

to develop and to use. The engineers and other personnel using these programs should have no need to learn difficult procedures.

- Hardware requirements.

The demands of orebody modelling, simulations, view changing and the massive amounts of data used by our computer programs could easily reduce the effectiveness of an incapable CPU. What hardware performance is needed and how can this be boosted?

- Resolution of CRT display

What resolution of display do we need? Resolutions of most popular monitors used for graphics today are 1280 x 1024, 1024 x 1024, 1024 x 768, 512 x 512 and 640 x 350 pixels.

A completed software product which display graphical images is usually well accepted by observers. The definition of an approach toward achieving a comprehensive mode of integrating software in graphics for the mining industry deserves much deliberation. Areas have been identified where computer graphics is applied in the mining industry and in light of these some of the expectations of hardware and software have been noted. The intricacies of hardware and software which allow the user to produce these images have not been examined yet. In the next section current computer graphics technology will be examined.

5.5 Computer Graphics Technology

5.5.1 Hardware

Computer graphics hardware can be categorized according to their function:

- Input devices
- Interactive devices
- Output devices

- Input Devices

Input devices are used to put pictorial data into the computer and to aid in the manipulation of this data. These devices are usually supported by software in order to perform these tasks.

Common input devices are:

The graphics tablet - also known as a digitizer. First a user defined origin is set up to be somewhere on the tablet, then a stylus is used to trace or draw an image on the tablet. The position of the stylus is converted to X, Y coordinates which represent the image. This device has been proven to be a fast and effective way of inputing pictorial data into a computer and is very popular.

The joystick - A stick which can be moved horizontally and vertically. The movement of the stick can be converted to X, Y

coordinates in order to control the position of the cursor on a screen. This device suffers the drawback of the user not being able to judge small, precise movements of the stick, thus positioning is difficult to judge.

The mouse - A device which is used very much like a joystick, however this instrument can be moved around on a flat surface since it is supported on wheels. A mouse is usually used to control the position of the cursor on the screen and to interact with a program through buttons on its top.

The light pen - This instrument contains a light sensitive device in its tip and is used mostly on vector refresh displays to return coordinates of luminated positions on a screen to the application program. Use of this device requires the user to hold the pen up to the screen for extended periods. This could be fatiguing.

- Output Devices

When an image has been created on a screen, output devices are used to produce a hard copy of this image on a permanent medium. Output devices include plotters, printers and cameras.

- Interactive Devices

Interactive devices refer to hardware which is used to view images in a mode which allows for easy modification, manipulation and recreation. The devices are predominantly CRT and flat panel displays.

CRT display types which are prominent today are vector refresh displays and raster scan displays. Vector refresh displays uses a CRT with a short-persistence phospor. All data are in the form of vectors which represent lines on the screen. This data may be modified in a selective manner as desired for image manipulation. With this form of display an electron beam strikes the phospor on the screen and causes it to glow. Because the phospor only glows for a short period, the picture must be re-drawn before the image fades. This requires rapid and frequent re-generation of images. Because of this a display processor is usually positioned between the computer and the CRT. If the image to be displayed is complex and the display processor not fast enough the displayed image will flicker.

Vector displays offer the following advantages and disadvantages:

Advantages include:

- High resolution of display
- Allowance for high levels of interaction through the use of devices such as the light pen.
- Varying levels of intensity
- Selective erasure
- Display hardware usually facilitates high transformational capabilities.

Disadvantages include:

- These display types are usually expensive
- Requires sophisticated hardware to generate and to prevent flicker with detailed images.
- Limited colour capability.

Raster scan displays generate images by having a beam sweeping the CRT screen line by line while it is being selectively turned on and off. Unlike a vector display which generate points and lines as single stroke operations, raster scans illuninates individual dots on the screen. These dots or picture elements are called pixels. These dots form a rectangular matrix on the surface of the screen and their size and matrix area determine the resolution of the screen. The computer stores information for each individual pixel. Colour on raster scan displays is obtained by having each pixel constructed with three adjacent phosphor colours red, green and blue. Illuminating these three colours to various intensities produces additional colours.

Disadvantages of raster scan displays include:

- Display resolution is limited by memory required to store information about each pixel.
- Transformation of images is difficult and requires sophisticated hardware and software support.
- Real time animation and image manipulation requires dedicated hardware support.
- Pixel resolution poses problems through offset when representing straight lines.

Advantages include:

- Complex images can be displayed without flicker because phosphor used has longer persistence.
- Polygons can be easily filled.
- Flexibility of operation allows for shading and other pixel attribute modification which produce images with more realism.
- The cost of these display terminals are moderate and is

decreasing with time.

A new form of display which has been increasing in popularity is the flat panel dislay. These fall into two categories; light-emitting diode displays(LED) and liquid crystal displays(LCD). These display types however, although colourful and robust are very slow at updating, thus they pose no threat to raster scan and vector refresh displays.

5.6 Software

The task of obtaining pictorial images on a graphics device requires the input of data, the manipulation of these data and the conversion of these data to forms which are comprehensible to that graphics device. Each stage toward the completion of this task requires the provision of computer software. Each input device requires specific software in order to interpret its data. Manipulation of this data through clipping, scaling, rotation, and translation of the images requires software. The conversion of data representing the final format of the image must be communicated to an output or interactive device, and that also requires software.

- 2-D Software

Data stored on a computer which represents a pictorial

image is usually defined by coordinates which differ from the coordinates required by output devices. 2-D graphical display of this image requires that the coordinates which define the image be converted to that of the display and that the image be clipped if only a window is required. Clipping refers to selecting all points within a specific window of the overall image, thus only the contents of this window would be transformed to a given viewport.

2-D graphics also involves the employment of line drawing algorithms and transformations such as scaling, rotations and translations. These mathematical algorithms all require the manipulation 2 and 3 dimensional matrix multiplication and can be CPU intensive.

- 3-D Software

3-D graphics is a result of the pursuit for realism in representation of pictorial images on a computer. Mathematically 3-D transformations are similar to 2-D, however as more lifelike images are produced the complexity of the algorithm increases. Previous approaches to computer graphics used wire-frame outlines to represent all objects. These forms are popular on vector refresh displays. Raster scan displays allow the introduction of colour in the display of the images. Thus algorithms for shading, creating shadows and depicting surface texture are often

employed. With 3-D graphics additional features are important image projection, sectioning, hidden surface elimination are a few of them. Complexity increases when blending techniques such as spline and Bezier curves are implemented. These algorithms all help add more realism to a display. However they require tremendous amounts of CPU attention in order to be effectively carried out.

The issues to be noted about the implementation of computer graphics algorithms on a computer are:

- The manner in which these algorithms are accessed to manipulate data and produce these images.
- The interface between these algorithms and the hardware devices employed.
- The real time performance of the hardware while constructing, modifying and recreating these images.

5.7 Software Standards

One of the most challenging problems facing designers of computer software involving graphics is the continuous influx and modification of graphics devices with which their applications must interact. Hardware innovation has resulted in features which were previously part of a device independent part of a general-purpose package being built into hardware. The development of graphics software which is portable and device independent will promote the industrial utility of these programs.

Throughout the 60s and 70s general-purpose packages have

become popular and extensively used. Although each of these packages provide device independence to some degree they each had independent specifications with respect to how the application program interfaced to libraries of graphics algorithms. Thus software became dependent on specific packages. Dissatisfaction also resulted because these packages often supported specific hardware and were slow to accommodate changes in hardware technology.

Ideally computer software which include graphics should be portable between different hardware environments without any need for modification of the interface to graphics algorithms. This software should also have the capability of utilizing any hardware device without extensive modification. On the same computer a device-independent application program needs to be compiled once. At execution time a user should be able to interchangeably select any particular device without recompiling the program.

Today the Graphical Kernel System(GKS) is an accepted international standard for the specification of hardware and software for computer graphics. GKS has gained prominence over its closest competitor - the Core Graphics System, by being accepted by the International Standards Organization(ISO).

GKS and CORE

International acceptance of GKS did not settle

controversy between supporters of GKS and CORE. CORE was developed by the Graphics Standards Planning Comittee(GSPC) of the US. GKS was developed by the German Standardization Institute(DIN). The major differences between GKS and CORE are:

1 - GKS is 2-D and CORE is 3-D. Work is currently in progress with ANSI and ISO on the implementation of 3-D primitives to GKS.

2 - GKS has standard language bindings, CORE does not. Because CORE became a pseudo standard without standard language bindings, each application programmer is faced with different interfaces to each CORE implementation.

3 - GKS has a bundled attribute concept(see below), CORE only allows the specification of individual attributes.

4 - GKS is accepted internationally and many hardware and software manufacturers are using this standard in the specification of these products because of world-wide market appeal.

At present GKS provides for greater universality of graphics-based software than CORE because of strict adherence of GKS implementations to the specifications provided by DIN, and its international acceptance.

An emerging graphics standard is PHIGS (Programmer's Hierarchical Interactive Graphics System). This standard is under development within ANSI.

PHIGS

PHIGS is a developing standard and is similar in concept to GKS. PHIGS however has 3-D and other high-performance

functions such as nested segmentation and segment editing. The concepts embodied in PHIGS serve sophisticated requirements for graphics implementations and allow for the support of state-of-the-art graphics displays. Today PHIGS is not widely accepted as a graphics standard. Although its functions extend beyond those of GKS it is not mature and is only now being proposed to ISO and ANSI as a possible standard. Commercial support for GKS exceeds that of PHIGS and there is much promise that commercial implementations of GKS will evolve as this standard is improved. This is forecast in light of the commitment already made by the software and hardware developers in Europe and the U.S. to the GKS standard.

The Graphical Kernel System

The GKS standard defines a set of functions which can be used to perform graphical tasks in a device independent way. GKS provides the following areas of functionality:

- Output primitives supported includes polyline,polymarker,fill area, text, fill array, and generalized drawing primitives.
- Multiple workstations are supported. These graphics devices may be accessed in an independent manner and these workstations may be allowed to utilize their local intelligence if present.
- Individually controlled, bundled and mixed attributes. A bundle may specify a primitive with a specific combination of attributes. This bundle can then be referred to by its bundle identification.
- Workstation dependent and workstation independent segments. A set of primitives may be given a segment name and manipulated as a single unit.

- User defined world coordinates may be normalized and transformed through GKS routines.
- Input devices are supported in six classes. These allow for interactive input of data. These devices may be operated in three modes in order to facilitate different synchronicity of input data.
- Inquiry features are supported so that an application program may inquire about the characteristics of different workstations and keep the program informed of particular hardware capabilities.

When programming with GKS the software developer simply includes calls to graphics routines in his/her application program.

For a developer of graphics-based application software, GKS offers graphics device independence, improved programmer productivity, extended software life and easier manageability. Software will have quick graphics functionality and allow for concentration of resources on application.

5.8 Integration of Software and Hardware

Although GKS provides for the universality of graphics-based computer software the performance of graphics hardware is still a major issue. In many instances the CPU lacks sufficient power to handle complex graphics tasks such as model rotation and perspective views. Performance boosting can be achieved in many ways; special-case software may be used, brute force methods of utilizing a faster CPU may be implemented or dedicated graphics

processors may be employed.

Graphics processors usually have high processing power, they incorporate routines in their firmware and execute them at high speeds to relieve application programs of repetitive tasks. These processors possess built-in text enhancement features, graphics routines for scrolling, zooming, filling and many other algorithms. Many graphics workstations today have on board graphics processors. Microcomputers which have in the past been past over because of their graphics capabilities are now looked upon more favorably because of the emergence of graphics processors such as Intel's 82786 Graphics Coprocessor and Texas Instruments' 34010 Graphics System Processor. Choice of a graphics processor is dependent on the computer used and is integrated in the architecture of that computer. The integration of hardware and software provides us with sufficient processing power to perform even the most demanding of graphics tasks.

5.9 Guideline for the Procurement and Development of Graphics based Software in the Ontario Mining Industry

Software

These guidelines are intended to satisfy the following criteria:

- software should be device independent
- software should be portable

- software should be easily maintainable
- during the development stage, software should have quick graphics functionality
- software should utilize and exploit available technology and be positioned for growth into tomorrow's technology
- graphics-based software developed at different sites should be easily integrated with each other
- all interface between application programs and graphics library functions should be consistent, regardless of programming language employed.

These criteria are all satisfied by the Graphical Kernel System within the scope of 2-D implenentations. Adaptation of the GKS standard provides us with a means of improving productivity, utility and lifespan of graphics-based computer software. GKS defines a library of user callable routines which may be integrated in an application program in a well defined and standardized way. Adherence to this standard will alleviate fears of application programs being "locked in" to hardware. Hardware selection for graphics capability will then be based strictly on performance needs, data storage capability, user comfort and cost. GKS provides us with a standard for the development of 2-D graphics based software in the Ontario Mining Industry. Because of the lack of both standardization and acceptance of proposed standards specifications of 3-D graphics implementations, no recommendations can be made to support any known system. It is recommended that 3-D graphics be implemented through the use of commercial tools packages which bridge the

pure standard and the real world practical applications. Commercial tools packages such as DI3000 and GK-2000 implement added value to recognized standards. Commercial packages should be considered with the following criteria in mind:

- The quality of the reference documentation and tutorial material accompanying the package.
- The functional extensions that go beyond the standard.
- The quality of technical support provided by the manufacturer.
- How well is the tools package designed to accommodate future changes to the standard e.g. 3-D to GKS.

With this in mind a package based on the GKS standard is recommended. The GKS base provides for compatibility with 2-D GKS implementations and the known support and commitment of standardization bodies and commercial developers worldwide.

Hardware

Hardware required to support software based on GKS has no specially desired specifications except of course that compatible GKS libraries and device drivers are available for the operating system. Hardware selection is to the most part based on cost and performance. The following presents possible hardware configurations for graphics.

Three common configurations are:

1 - Host computer and graphics terminals
2 - Graphics workstations
3 - Graphics oriented desktop computers - personal computers.

As technology changes the distinctions between these configurations grow less clear. The difference between graphics workstations and terminals are that workstations do significant amounts of local computations while terminals do not. Graphics terminals must always be attached to a host computer while workstations may be stand-alone systems. Because of the stand-alone feature of graphics workstations they are often thought of as personal computers. These machines however host much more powerful CPUs, more sophisticated architecture, and cost up to 20 times as much.

Graphics workstations usually have higher resolution screens than terminals and carry on-board graphics processors which greatly boost performance. These systems communicate data to the CRT at much faster rates than terminals, since terminals are usually connected to their hosts with RS 232 interfaces which have a maximum communication speed of 19,200 baud. Because of their communication restriction terminal updates are much slower than workstations.

Competing directly with workstations are PCs, since most workstations today are configured with 32-bit microprocessors and PCs are now appearing with CPUs of the same performance. New graphics processors for PCs provide almost comparable graphics performance as workstations at significantly lower costs. Table 11 gives a comparison of harware configurations and costs.

HARDWARE	COSTS($)
Graphics Terminals	1,300 - 9,000
Graphics Workstations	15,0000 - 60,00000
Personal Computers	6,000 - 15,000

TABLE 11: COMPARISON OF HARDWARE CONFIGURATION AND COST.

Of significant importance in the selection of hardware for computer graphics applications is the resolution of the display monitor. As was previously mentioned, resolution of a monitor is measured by the size and matrix area of the pixels on the screen. Screens of higher resolution allow for the representation of greater detail in the presentation of graphical images. This also enables easier interpretation of the images portrayed by the computer and improves man-machine interaction. The selection of a screen of a specific resolution thus lies with one's particular needs with respect to graphical detail, subject to the constraint of user comfort. The cost of monitors increase with their graphical resolution.

Conclusion

Graphics-based applications of computer software in the Ontario mining industry greatly improve productivity. These programs however need to be developed within certain specifications to improve their overall utility. These issues of utility focus on device independence, portability, maintainability and ease of development.

The Graphical Kernel System satisfies these requirements and has been proposed as a standard for the development of 2-D graphics-based software in the Ontario mining industry. A commercial tools package based on the GKS standard is recommended for 3-D implementations.

CHAPTER 6

DATABASE SOFTWARE

6.0 INTRODUCTION

A database is a fully integrated collection of files brought together to serve multiple applications.

Data is usually accessed via a database management system which allows the same data to be viewed in a number of different ways. The use of a database management system is shown in figure 13.

6.1 The Database Management System

The Database Management System is a collection of software and hardware components that enables individuals to process large amounts of associated data. To do this, the database management system must be able to configure a computer system with secondary storage so that it can define, create, store, update archive and manage integrated data files. Furthermore, many application programs should be able to use the data at the same time.

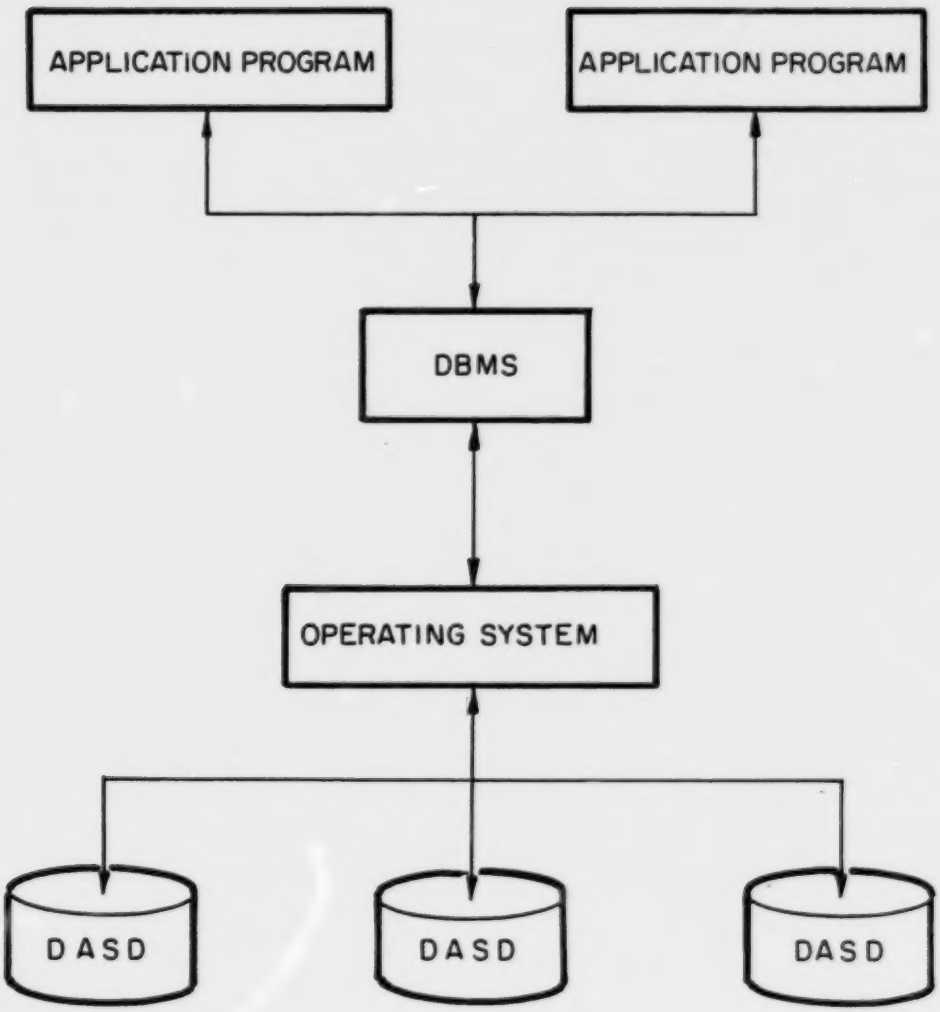


FIGURE 13 : USE OF A DATABASE MANAGEMENT SYSTEM (DBMS).

The following are the advantages of using a database management system:

1. More information can be stored in the same amount of space in a database than in a file system.
2. Less data redundancy can be obtained by storing associations between data values.
3. Database management systems provide a greater degree of independence between program and data.
4. Elimination of conflicting data due to update of all relevant data.

However there are disadvantages associated with the use of Database Management Systems. Database Management Systems are more vulnerable than file-based systems because of system failures. It is hard to determine the state of a database before failure, especially in multi-user systems. Moreover, the sophisticated programming in a Database Management System requires longer design and implementation time for new applications. Modification of existing applications also takes more effort.

6.2 Components of a Database Management System

The components of the Database are shown in figure 14. The architecture is composed of hardware (storage devices) and software (all other components). All applications interface to a functional database. The functional database is the application

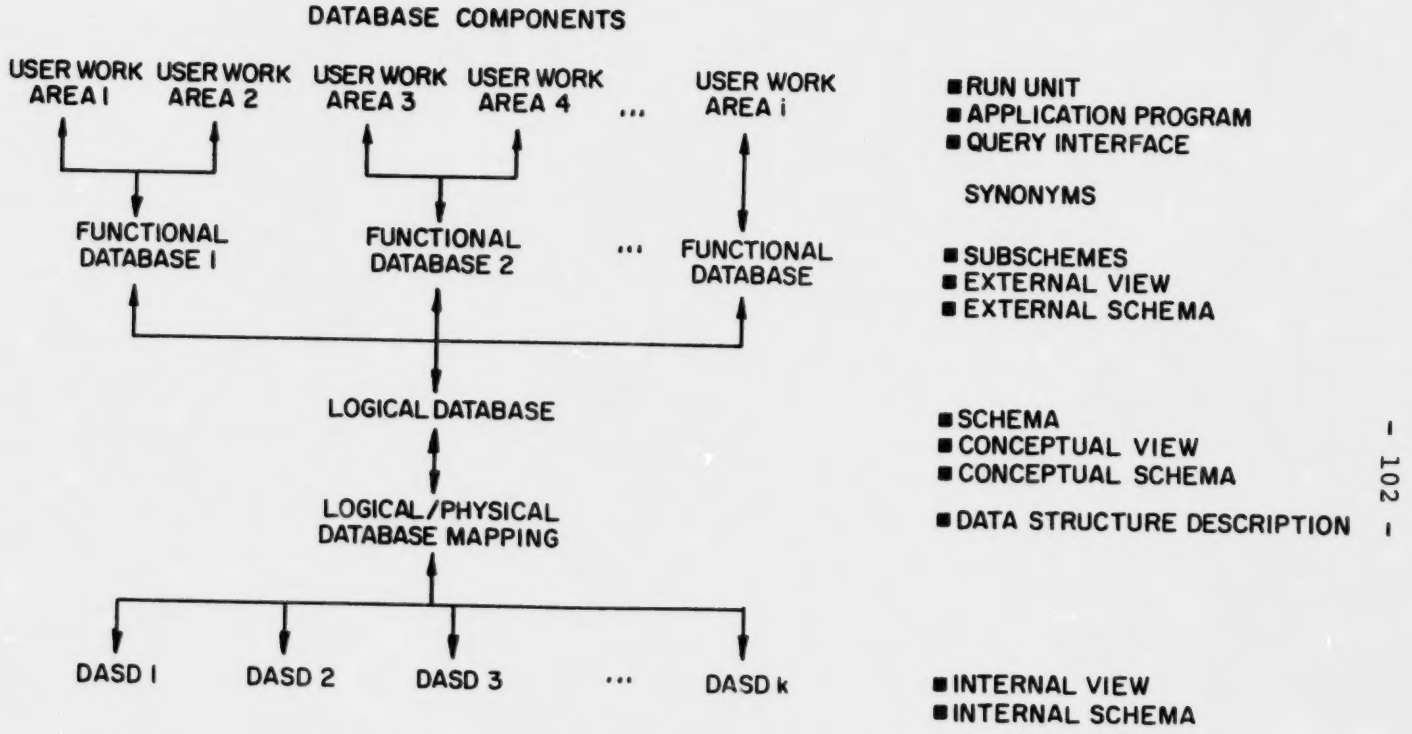


FIGURE 14 : COMPONENTS OF A DATABASE MANAGEMENT SYSTEM. AFTER SMITH AND BARNES

program's (or user's) view of the database. The functional database communicates with the logical database which can translate a request, so that the logical/physical database mapping module can determine the actual storage location of the data to be accessed from a storage device.

A database management system is created based on the principles of logical and physical data independence. Logical independence means that changes to the logical database structure have a minimal effect on the application programs. Similarly physical data independence means that changes in the physical storage do not affect the logical database.

6.3 Specifications for Database Management Systems

The life cycle model of the database software follows the same general rules set up in the second chapter. In the present chapter a few additional points will be discussed.

During the design phase it is absolutely necessary that the designer of the database, in addition to flow charts, uses data flow diagrams. These diagrams should be part of the documentation of the software and become available to the agency responsible for the software maintenance. Moreover the diagrams should be reviewed during the review meetings (preliminary design and final design reviews). General data flow diagrams are an excellent tool for the communication between the customer and the programmer.

The Database implementation stage involves writing the code for the database components, loading the database and compiling application programs to run against the database.

Database Maintenance involves monitoring a working database system and recovering from a failure.

During monitoring, the database is checked and corrected in case of anomalies in application program performance. Moreover, in case of a sophisticated multi-user control database, the database performance is monitored with utilities that provide statistics on storage utilization, access time and general system throughput. When database access performance degrades to an unacceptable level, the maintenance staff reorganizes the physical data base and installs a new logical/physical mapping module.

Recovering from a failure is a difficult task. This is a major disadvantage of all database management systems. There are two strategies for recovering: archiving and rollback.

Archiving is the simplest way. It is recommended for small databases with a small number of users. In this method the database is copied periodically. After a failure, the database is restored from the most recent copy. The disadvantage of this method is the time required to re-enter the new data.

Rollback begins with the database in its soiled state and attempts to undo all transactions that did not complete before failure. However, this is a more expensive method because it requires more sophisticated programming and skilled personnel.

This method is recommended in the case of a multi-user database with frequent changes by several people.

6.4 Database Administration

In order to utilize a database management system effectively there must be a person or group that looks after the system and makes decisions about its control. Small databases with few applications and users have fairly simple requirements (one person). Commonly, such databases have one user at a time. In case of large systems supporting concurrent users, the administration is the concern of the database administrator. The database administrator is a group of people responsible for designing, creating, loading and maintaining the database. The database administration is a centralized source for information about the current state of the database for the application programmers and users. It also serves to maintain data integrity, enforce security precautions, minimize redundant data and back up the system.

6.5 The Mining Industry and Database Systems

Databases are used extensively in the Mining Industry. However, in many cases "databases" are single file systems used only for a single purpose, and usually by a single programmer.

Database systems are used in the following areas:

1. Maintenance. Records of each piece of equipment are kept so that repair data, activity, production data associated with the particular item, component data (change out histories and inventory) are available at any point. Up-to-date information can be used for the equipment evaluation, simulation studies and purchase decisions.
2. Inventory. Database programs search for various items, identify slow moving items, prepare reorder listings, invoices and purchase orders.
3. Accounting. Database management systems are used to track the costs, prepare the payroll and do the accounting.
4. Drilling - Blasting. Database programs keep record of drilling bits, performance of bits, rock data, drilling costs, blasting patterns, explosives loaded, explosive performance.
5. Exploration. Data from the geological surveys and drilling can be used in databases. The databases can be used to create the orebody model which can be used for mine planning and geostatistical analysis.
6. Surveying. Surveying data can be stored directly in a database which can be used for mine planning, blasting, and ventilation for the underground mines.

It is obvious that a central database would be of particular interest because the same data are used by a variety of users. This database would be the most up-to-date information available and would minimize the possibility of conflicting data.

Moreover data redundancy would be minimized, since not every user would have to store the same data.

6.6 Database Storage Requirements

The database storage requirements are a function of the amount of information handled. A typical size of a hard disk for a centralized database can be as large as 150 Megabytes.

A typical size for a Maintenance Module for the Mining Industry is 10 - 20 MB.

It should be noted that the storage devices must be direct access storage devices such as hard disks or drums. Tapes can be used only for archiving since they are sequential storage devices.

CHAPTER 7

7.0 COMPUTER HARDWARE REVIEW

This chapter deals with a review, of microcomputer related hardware. It begins by reviewing the internal functioning of the microcomputer. A general understanding of the microcomputer is required in order to develop an informed purchasing strategy. After reviewing the fundamentals of the microcomputer the various hardware items are discussed. This includes input/output hardware as well as modems and various storage media.

7.1 The Internal Functioning of a Computer

The computer and all the internal functioning is based on electricity. Therefore, it can recognize only two states, on and off. It is evident then, that the computer must use the binary system to perform all computations. In the binary system, off is represented by a 0 and on is represented by a 1. Each character or number must thus be encoded as a group of 0's and 1's.

Each binary digit (0 and 1) is stored in one position in memory, that is called a bit. To represent the number 1 in binary a number 1 is used, to represent the number 2 a 01

combination is used. The number 1 is represented by 1 bit of memory space where as the number 2 is represented by 2 bits of memory space. All the positions in memory follow one another in succession without dividers, and if the memory contents were randomly examined a series of memory will look like "10001011101". In order to differentiate between two numbers a series of either 4, 8, 16, 32, or 64 bits is used to define one number. The word byte is used to refer to a sequence of 8 bits.

Figure 15 is representative of the five functional units of a computer system. Each component is a necessity to the functioning of the unit as a whole.

The central processing unit is composed of the control unit and the arithmetic and logic unit. The arithmetic and logic unit performs arithmetic manipulations and logic operations on data passing through it. Arithmetic manipulations would include addition and subtraction and logic operations would consist of such logical operations as AND and OR.

The control unit performs the function of fetching, decoding and executing successive instructions stored in memory. It is the most fundamental piece of the entire computer system because it times signals and data transmition throughout the computer system.

The memory unit is used to store data, programs and other information. Memory has been disected into two portions, the main memory and the mass memory. The main memory is completely accessible to the control unit and is used to store

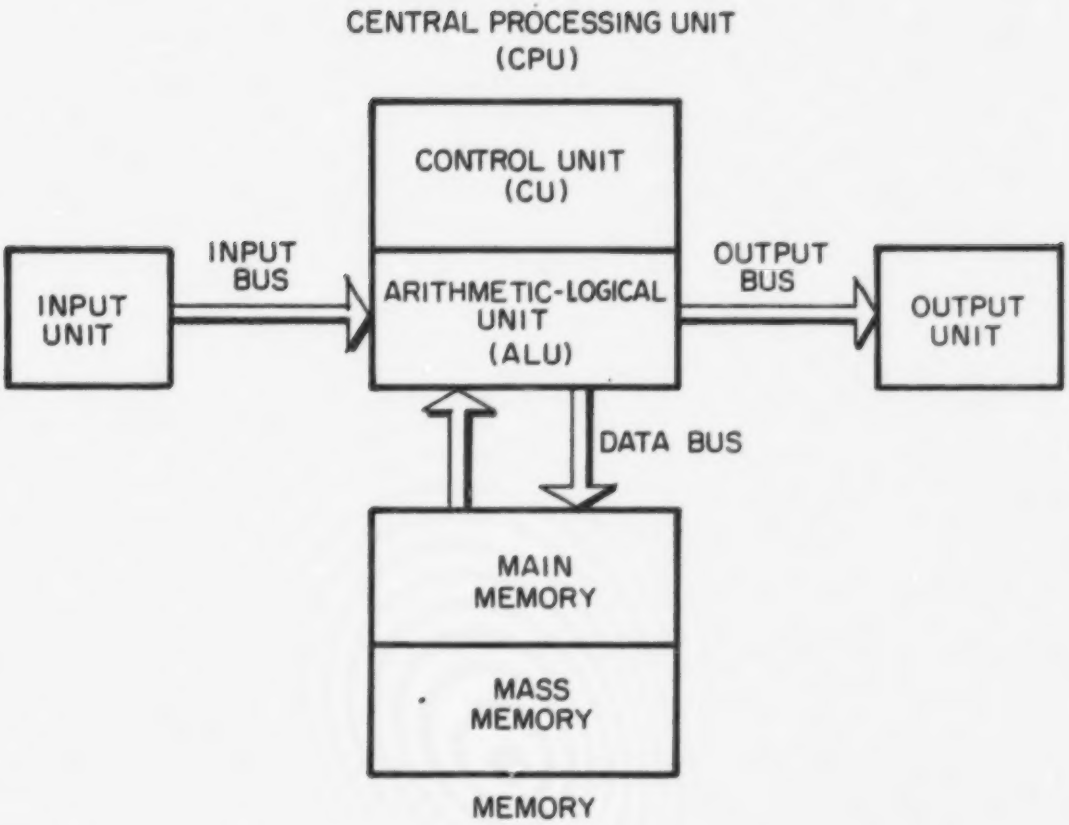


FIGURE 15 : THE FIVE FUNCTIONAL UNITS OF A COMPUTER SYSTEM.

information during manipulation. This type of memory must be quickly addressable so as not to slow down the central processing unit. A typical cycle time for the main memory is in the order of several hundred nanoseconds. The maximum size of the main memory is usually limited by the addressing capability of the central processing unit as will be discussed. The main memory is usually in the form of computer "chips".

The mass memory is used to store programs or data that are not currently required. Accessing the mass memory usually slows down the computer. For this reason, the main memory should be large enough that frequent access to the mass memory is not required.

Mass memory is in the form of hard disks, floppy disks, tape drives and optical disks. This mass memory is basically a tremendous number of electromagnetic switches or in the case of the optical disc, depressions.

The other notable difference between main memory and mass memory is that main memory is generally volatile memory. Volatile memory is memory that loses the contents of its registers when the power supply is interrupted. Mass memory, or non-volatile memory, on the other hand, does not depend on a continuous power supply in order to maintain its registers' contents.

Two types of main memory must be considered; random access memory (RAM) and read only memory (ROM). ROM is generally used to store permanent information that the manufacturer has

deemed necessary for the efficient functioning of the machine. RAM on the other hand is memory that can be written to, or read from, and is changed continually during the life of the computer.

The input and the output units are used for communication with hardware other than the computer's central processing unit. The input unit communicates with items such as the keyboard, a mouse or a digitizing board. The output unit communicates with items such as the monitor, a plotter or a printer.

All information is transmitted from one unit to another via buses. Buses are information transmitting channels. The data bus transmits data between memory, the arithmetic and logic unit and the control unit. The address bus is used to transmit information as to where a particular piece of data or a piece of program is located in memory. The control bus synchronizes the system.

Bus sizes are measured by the number of bits of data that may be transmitted simultaneously. A 16 bit bus may transmit a 16 bit piece of data in one machine cycle.

Registers are storage locations inside the processor. Registers in the control unit are used to keep track of the overall status of the program currently under execution. Control unit registers store such information as the current instruction, the location of the next instruction to be executed, and the operands of the instruction. In the arithmetic and logic unit, registers store data items that are added, subtracted,

multiplied, divided, and compared. Other registers store the results of arithmetic and logic operations.

An important concept that affects both the speed, precision and performance of the processor is the size of the registers. Technically the term "word size" describes the size of an operand register. The term is often used more loosely however to describe the size of the pathways or buses to and from the processors. The current range of word sizes in general purpose computers range from 8 to 64 bits.

The amount of addressable memory or memory that may be readily accessed by the processor is dependent on the register word size. Each location of data in memory has an individual address number. As the processor uses binary to count, the base 2 system can be used to count memory address spaces. A 4 bit address system would be limited to 2^4 or 16 addresses or memory locations. An 8 bit address system would be limited to 2^8 or 256 memory locations. A 16 bit address system limited to 2^{16} or 65, 536 memory locations and so on. In order to remove the restriction of an 8 bit processor being limited to 256 memory locations, the 8 bit processor may have a 16 bit memory address architecture. This involves addressing each memory location using two 8 bit addresses. Thus an 8 bit processor can address 65536 memory locations. However, by addressing memory in this fashion two address cycles must take place, thus significantly affecting the speed of the operation.

Memory is generally referred to as the number of bytes or 8 bit sequences available. Because memory size is generally millions or even billions of bytes, the terms kilobyte, megabyte and gigabyte are used. One kilobyte of memory is the equivalent of 2^{10} (1024) 8 bit memory locations. 64K of memory is thus equal to 65536 8 bit memory locations. A 32 bit machine can address 4 gigabytes. As a rule of thumb 64K of memory is roughly equal to enough storage capacity for 10,000 words.

One measure of processor power is clock speed. Clock speed is measured in megahertz (MHz). Each processor operation proceeds in fixed time intervals. Each operation, such as fetching an instruction, thus takes one or more cycles to complete. If the processor's clock speed is 8 MHz as opposed to a 4 MHz clock speed and the processor architecture is the same, then the 8 bit clock speed processor will perform the operation twice as fast. However, clock speed is not usually a good measure of speed when comparing different processors. Different processors use different architectures, and as a result, the processor running at 8 MHz may require twice as many operations (or clock cycles) to complete the same task. The end result is that both processors may perform the operation at the same speed.

Performance of microprocessors is generally dependant on the number of bits a processor is capable of working with, at one time. An eight bit processor can handle an eight bit piece of data in a given time period. A 16 bit processor can handle 2

bytes in the same unit of time, whereas, a 32 bit processor can deal with 4 bytes. If program segments dealing only with calculations are involved, the execution of the 16 bit processor may be two times faster than the 8 bit processor, due to the manipulation of twice as much data in the same time period. Also, the 16 bit chip may have a larger and more intricate instruction set than the 8 bit processor, and operation may require fewer machine cycles to complete. One example is the 8 bit processors method of multiplication. Because of its small instruction set, the 8 bit processor must do multiplication by successively adding the numbers together. However, the 16 bit processor incorporates multiplication in its instruction set. It is not difficult to imagine the difference in processing speed due to this enhancement alone.

As indicated, when the processor is dealing strictly with calculations, the 16 bit machine is at least twice as fast as the 8 bit. However, when processing program segments involving input/output, logic comparison, and storage retrieval operations, the 16 bit machines may have less of an advantage. In general, the 32 bit machine will be faster than the 16 bit which will be faster than the 8 bit machine.

When dealing with operations that are not particularly memory intensive, the speed and precision of the hybrid 8 bit and 16 bit microprocessors provide excellent service. However, many engineering and scientific applications are memory intensive and require a high degree of accuracy. As an example examine the

high resolution graphics terminal. Displays or cathod ray tubes (CRT) consist of the display screen being divided into thousands of elements (pixels) that are turned on or off or to varying colors in between by the computer. Graphics screens might be composed of an array of 1024 X 1024 pixels. Each pixel may have the ability to represent 10, 100 or 1000 different colors and shades. Ten different colors would require 1024 X 1024 X 10 = 10,485,760 pieces of information or 10 Mb to represent one single display screen. This represents a tremendous amount of data to be both processed and stored. Finite element analysis may require a similar amount of memory. It is obvious that for these types of computer applications many restrictions are imposed by using 8 bit or 16 bit processors. It is for this reason that these types of engineering and scientific programming have been restricted to the minicomputers and main frame computers in the past.

More recently, however, microprocessor manufacturers have developed true 32 bit microprocessors. It is evident that as these processors become introduced into more and more micro computers, engineering and scientific programming will be done more and more on the micro-computers. Certainly by the 1990's, even the most powerful engineering and scientific programs the mining industry is interested in, will run on micro-computers.

7.2 Computer Storage

The computer is a machine used to process information. In order to do this, both the computer code used to manipulate data and the data must be stored in some device accessible by the computer. Many types of storage are available, which vary in both cost and speed of accessibility to the machine. Figure 16 shows the storage hierarchy pyramid. A faster access time is obtained by moving up the pyramid. A larger storage capacity and lower cost per bit stored are the result of moving down the pyramid.

Cache storage is CPU storage that is used during processing operations by the CPU. This type of storage is by far the most expensive. Cache storage is used as a "scratch pad" type storage and is used to temporarily store instructions that may be used many times during the execution of a program. This tremendously increases the processing speed of the CPU. This type of memory was once only found in mainframe computers. However, it is now being introduced to mini and micro computers. This may well be one of the largest single differences between these three groups of computers as far as speed of execution and cost are concerned.

Primary memory or CPU storage is the second fastest memory to access. This type of storage is generally of the semiconductor type. This type of storage while expensive just a short time ago, is decreasing in cost by about 50% every three

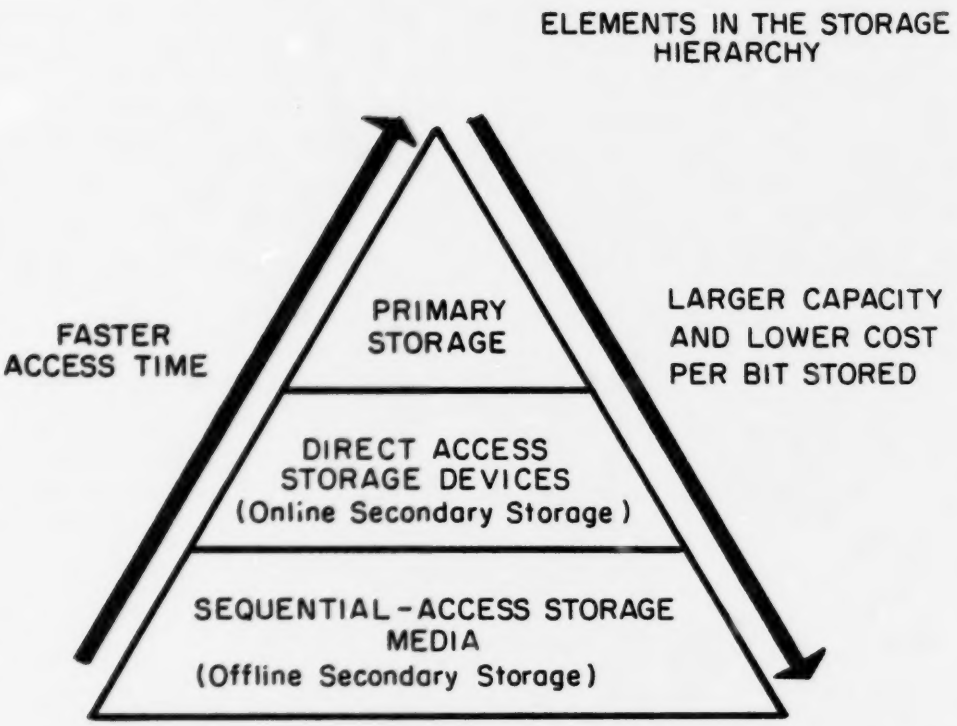

ELEMENTS IN THE STORAGE HIERARCHY
FASTER ACCESS TIME
PRIMARY STORAGE
DIRECT ACCESS STORAGE DEVICES (Online Secondary Storage)
SEQUENTIAL-ACCESS STORAGE MEDIA (Offline Secondary Storage)
LARGER CAPACITY AND LOWER COST PER BIT STORED

FIGURE 16 : THE STORAGE HIERARCHY PYRAMID.

years. Today primary memory is available for approximately $500.00 per megabyte. Figure 17 may be used to show the decrease in cost over the past 20 years. Another significant difference between micro computers and main frames is the amount of primary memory, with its shorter access time relative to secondary memory.

Primary memory is often referred to as random access memory (RAM). Primary memory is generally limited by the following:

1) cost
2) CPU addressability
3) CPU design
4) operating system limitations.

Primary memory size is generally an option in the selection of a microcomputer. For reasons of cost of primary memory, purchasers often select not to order their machine with its maximum capacity of primary memory.

Primary memory can also be limited by addressability. Each byte in the memory must be unique in terms of addressability. Thus the number of bytes of memory is limited by the largest number the CPU can form.

Primary memory is most often limited by the design of the CPU (the size of the address bus) or by other hardware limitations. The function of the address bus and the effects of its size have already been reviewed. An eight bit address bus

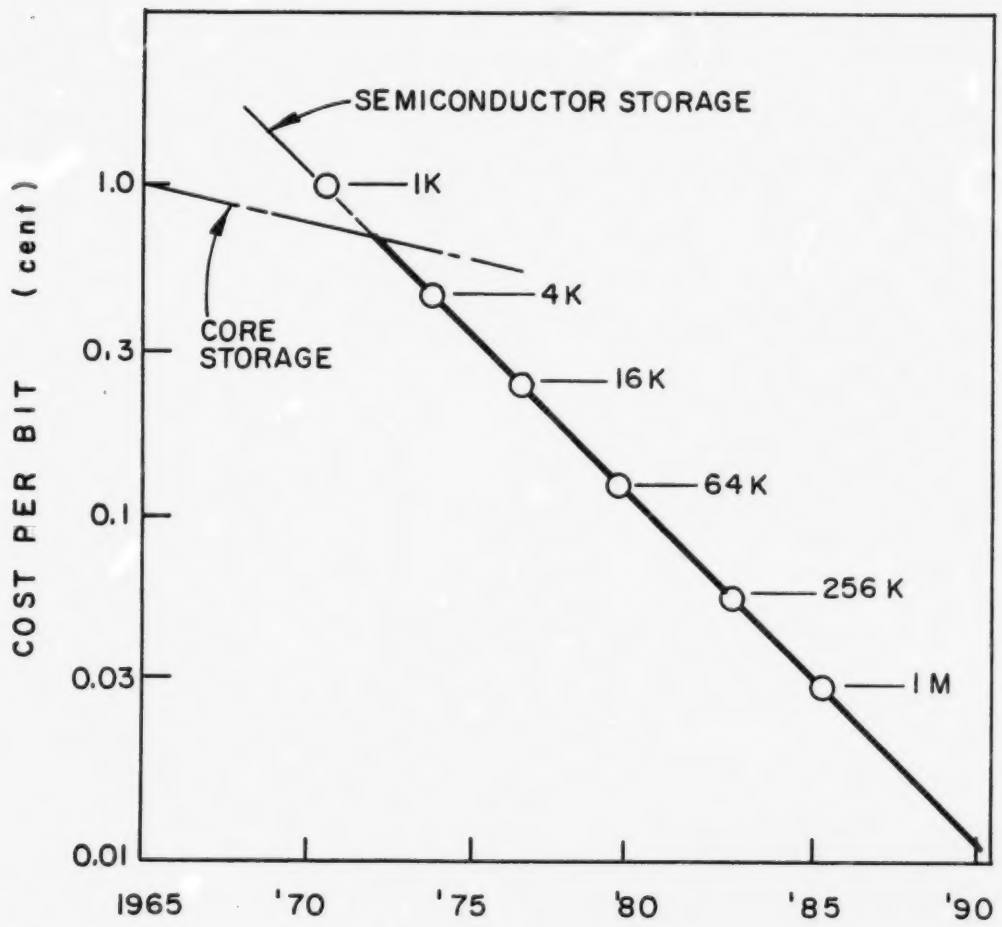


FIGURE 17 : THE NUMBER OF BITS THAT CAN BE STORED ON A SILICONE CHIP HAS QUADRUPLED EVERY THREE YEARS, AND THE COST PER BIT OF STORAGE HAS BEEN CUT IN HALF EVERY THREE YEARS. THIS TREND IN SEMICONDUCTOR STORAGE IS EXPECTED TO CONTINUE.

can only address 64 kilobytes of memory while a 32 bit address bus can address four gigabytes of memory (2^{32}). Limitations can also be imposed by other pieces of the hardware and while the address bus may be capable of addressing 4 gigabytes of data, other hardware components may limit the machine to a significantly smaller amount of memory.

The operating system selected by the machine operator can limit the amount of primary memory usable at any particular time. MS-DOS for example is limited to working with 640 kilobytes of primary memory at the current time.

SECONDARY MEMORY

Secondary memory, or mass memory, is for permanent storage of programs or data, and is not lost from the memory unless specifically erased. It is much less expensive than primary memory, however, it suffers from a longer access time.

Magnetic tapes were one of the first forms of mass storage. The tapes are coated with a thin layer of a metal oxide. The tape may be magnetized in a pattern interpreted by the tape drive and the computer. Magnetic tapes are still a very common form of mass storage. In micro computers they are used for backing up disk files. A 12.5 mm wide tape has a recording density of approximately 12,500 bits per centimeter. Reels of

tape are commonly 750 meters in length. Thus a tape of this nature may sort approximately 120 megabytes of data.

Reel to reel is the oldest form of tape drive. Reel to reel drives have been used in conjunction with main-frame computers for many years. The data stored on these tapes can be read many times without destruction. The tapes can also be used repeatedly for storing new data.

Other types of tape drives have also been developed that function in a similar manner to the reel to reel tape drive. Streaming tape drives, cassette tape drives and other forms of miniature tape drives are available. All these forms of tape drives have proven reliable for storage of large volumes of information. Streaming tape drives have been widely accepted for reasons of cost (a 20 Mb tape drive can be purchased for under $1,000.00) and efficiency by micro-computer users as a backup method for disk files. The single largest drawback of the tape type file systems is the long access time required to retrieve stored information.

The disk storage format was developed in order to decrease the access time for data retrieval. Retrieval of stored data from disk type storage can be thousands of times faster than from tape. This fast retrieval time also allows disks to be used as slow RAM or primary memory for some applications. Disk storage is available in two major formats, the "floppy" (flexible) disk and the hard disk.

The floppy disk system was introduced by IBM in 1972 as a faster retrieval device than the tape drive. Floppy disks have become one of the most popular storage mediums for micro-computers. They are available in sizes of 3 1/2" and 5 1/4".

Storage capacity of the floppy disks can very significantly, from 125 kilobytes for a single sided single density disk to 1.2 megabytes for a double sided, high density disk. Advantages of floppy disks include:

1) a standardized format and design
2) their small size that makes them easy to handle and store
3) capability of direct access to data stored on them
4) inexpensiveness.

Floppy disks, however, suffer from several disadvantages. They are easily damaged by heat, humidity or foreign materials. The rate of data transmission is slow relative to hard disks. They wear out with use due to the actual contact of the read-write heads with the disk. And they are susceptible to erasure or partial erasure of the contents when placed in close proximity to any type of electrostatic charge.

Hard disks by their name imply that the stored data resides on a hard or rigid disk. The disk is coated with a magnetic oxide material. Large storage capacity is achieved by stacking many disks above each other and by using many read-write heads. Hard disc capacities range from 5 megabytes in size for

smaller units to over 1 gigabyte in size for the larger units. Sizes range from under 4" to over 14 inches with the most common size for micro computers being the 5 " size. A 20 Mb hard disk can be purchased for under $1,000.00.

A hard disk allows for the storage of many programs without switching floppy disks. The largest single advantage of the hard disk is the much faster access time (20 times) than floppy disks.

A relatively new data storage medium and quite possibly the storage medium of the future is the optical disk. Rather than using a magnetic media for storage, the optical disks use a laser to burn a tiny pit or indentation into the disk coating. Data retrieval is via a much less powerful laser that reads depression or non-depressions in the disk covering surface.

While magnetic media storage devices are reaching their density limitations, optical storage devices will allow tremendous growth. The data may be stored in a much more compact format. One-hundred gigabytes of data may be stored on an optical disk the same size as a magnetic disk capable of storing only 300 kilobytes of data.

The cost of the optical media is approximately one-tenth the cost of magnetic tape and about one-hundreth the cost of magnetic disk. Another advantage of the optical storage method is the ability to access data much quicker than for the magnetic media storage methods.

One major disadvantage of the optical storage media, the inability to erase the stored data. Thus, data cannot be written over previously stored material and so the optical disks cannot be reused.

Another disadvantage, which is changing with time, is the cost of the actual disk player. A player capable of writing to the optical disks can cost more than $50,000.00. For many users this cost is prohibitive. However, writing to the optical disks is provided by service companies. Devices to read from the optical disks are available at costs between $1,000.00 and $10,000.00.

7.3 Microprocessor and Micro-Computer Selection

Microprocessors have advanced from the grandfather of microprocessors, the 4 bit Intel 4004 in 1971 to the stage of 32 bit microprocessors such as the Motorola 68020 and the Intel 80386 today. In this 15 year time span, the microprocessor has progressed from the processing element of a desk calculator, to the processing element of a powerful computer. The processing power of a microcomputer today, is comparable to the processing power of a mainframe computer just 5 years ago.

The main focus of this chapter is not to identify one particular microprocessor as being the "best", but rather to review the progress that has been made in the past several years,

and outline the perceived direction in the next several years as pertains to engineering and scientific programming.

An examination of Table 12 reveals the steady progress of several of the leading microprocessor manufacturers. The first generation of microprocessors was the four bit processors. These processors suffered from both a lack of speed and memory addressability. The 8 bit processors were introduced next. The 8 bit processors also suffered from the same lack of speed and memory addressability. The various microprocessor manufacturers solved these problems in a variety of ways.

Many of the microprocessor manufacturers increased the amount of addressable memory by using a two byte or 16 bit addressing format. By doing so, the addressable memory was expanded form 256 bytes to 64 kilobytes. Intel, realizing a need for greater then the 64 K addressable with a 16 bit address went one step further to a segmented type architecture for the Intel 8088 microprocessor. Intel's scheme relies on the 8088's recognition of memory being composed of segments, each segment being 64 K in length. In order to identify the specific 64 K segment in the memory, segment registers were needed. Intel designed the architecture of the 8088 around a 4 bit segment addressing format. Thus, with 4 bits to locate the segment and 6 bits to locate the specific address within the segment, a total of 20 bits are used. This allowed 2^{20} (1 megabyte) bytes of

COMPANY	MICROPROCESSOR	WORD SIZE (bits)	ADDRESS BUS (bits)	I/O DATA BUS (bits)	MAX RAM	CLOCK SPEED (MHz)
ZILOG	Z-80	8	8	8	64KB	4
MOSTEK	6502	8	8	8	64KB	4
INTEL	8085	8	8	8	64KB	6
INTEL	8088	16	20	8	1MB	8
INTEL	8086	16	20	16	1MB	10
ZILOG	Z8000	16	24	16	16MB	10
NATIONAL	16032	32	24	16	16MB	10
MOTOROLA	6800	32	24	16	16MB	12.5
INTEL	80286	16	24	16	16MB	8
MOTOROLA	68020	32	32	32	16MB	12.5
INTEL	80386	32	32	32	4GB	10

TABLE 12: A COMPARISON OF SEVERAL 8-, 16-, AND 32-BIT MICROPROCESSORS SHOWING ARCHITECTURAL DIFFERENCES IN THE BUSES, WORD SIZE, AND SPEED.

memory to be addressed; the only restriction was that memory had to be addressed in 64 K segments.

Soon after the hybrid 8 bit chip was developed further refinements allowed the manufacture of true 16 bit chips. These chips were developed around a 24 bit addressing architecture and could thus address a full 16 MB of RAM. This increase in access to larger amounts of memory had two input/output advantages:

1) it permitted the use of more sophisticated and powerful programs and
2) it increased the rate at which the processor could run these programs.

Traditionally engineering and scientific applications have required the most powerful state of the art computers. Personal computers were generally regarded as being too restrictive until the introduction of the 16 bit PC in the summer of 1984. The 16 bit model removed many of the restrictions from the personal computer. Memory addressability was increased from 1 MB to 16 MB, thus partially removing this restriction. Processor speed was increased as well.

Further refinements consisted of the development of a true 32 bit microprocessor. This processor has the ability to process data 32 bits wide, address 32 bits simultaneously along the address bus, and allows the addressing of 2 or 4 gigabytes of data.

By 1990 tens of thousands of bytes of high speed cache memory could be incorporated on a microprocessor chip. Microprocessors will function using the same instruction sets as mainframe computers. This will make the microcomputer much more powerful and will allow programs designed on mainframes to be directly portable to the microcomputer. By 1990 a typical microcomputer will have greatly increased storage capacity and will perform most of the functions of today's mainframes. Yet, it will be compact enough to occupy the corner of an office desk.

The rate at which microprocessor and computer technology are advancing, make it futile to specify or select one particular processor or computer type. Rather, a list of microcomputer specifications for many of the microcomputers on the market today, as well as the previous reviews of the internal functioning of the microcomputer and the support hardware, allow one to develop an individual guide to microcomputer selection.

Rule of thumb costs have been developed for a range of computer powers.

Workstations

Of particular interest to the mining engineering community are the workstations. With the recent developments (32 bit microprocessors), it is obvious that personal computer capabilities are now reaching the workstation level.

Briefly, a workstation must provide all the computing tools that engineers need in their daily work routines. This includes specialized applications like CAD and large scale numerical modelling, as well as general purpose tools for programming, writing and sharing information over networks.

Workstations offer true multitasking capabilities and provide enough computing power so that the system performance is not degraded because of multitasking.

The CPU has a 32 bit address space and an execution speed of at least 1 million instructions per second (MIPS).

It must be noted that the multi-tasking environment requires several megabytes of main memory and a virtual memory system based on disk storage and network resources.

It seems that there is an overlap of performance between low end workstations and high end personal computers. However, one cannot judge only from clock speeds, processor MIPS and benchmark results. The difference shows up when multitasking is encountered. Workstation performance is optimized by balancing of CPU and operating system throughput.

The most distinctive characteristic of workstations is their graphics capabilities. Displays are as large as 19 inches diagonally, allowing several windows to be displayed at once. The lowest acceptable screen resolution is 1024 X800 pixels. Some workstations go as high as 1280 X 1024 pixels. Color systems can display at least 256 colors simultaneously from a palette of more than 4096 shades. Graphics speeds range from

5000 two dimensional vectors per second to 40,000 vectors per second when a graphics accelerator is used.

Another important feature of workstations is their networking capabilities. Workstations communicate in local area networks at speeds up to 10-12 Megabits per second. This is 10-20 times faster than the speed at which personal computer networks perform. The high speed makes it practical to share memory resources through network traffic as a part of the virtual memory scheme, giving workstations virtual memory space of 64-256 Megabytes per process. Another advantage of workstation networking is the ability to distribute processing to other systems in the network.

The price for the basic workstation ranges from $30,000 - 50,000. On top of this cost one will have to add the cost of the software and additional hardware to upgrade the system.

Typical costs are:

RAM expansion	$1000 per Mb
Hard disks	$7000 for a 70Mb disk $11000 for a 150 Mb disk
Software	$3000 per compiler $1500 for the GKS system

It must be mentioned here that hard disk memory requirements can be extensive in the case of workstations. In order to have UNIX, various compilers and the applications software on the hard disk it is recommended that a hard disk of a capacity of more than 140 Mb is purchased with the system.

32 Bit Microcomputers

Basic Version -1 Mb RAM
-1.2 Mb floppy disk drive
-40 Mb hard disk drive

Upgraded version -10 Mb RAM
-1.2 Mb floppy disk drive
-130 Mb hard disk drive

These machines range in price from a low of approximately $10,000.00 for the base machine to a high of approximately $18,000.00 for the upgrade.

16 Bit Version

Basic Version -256 K RAM
-360 K floppy disk drive
-no hard disk

Upgraded Version -2.2 mb RAM
-1.2 mb floppy disk drive
-70 mb hard disk drive

These machines range from a low of $6,000.00 for the basic machine to a high of approximately $12,000.00 for the upgraded machine.

The above mentioned costs will be higher when software and high resolution color monitors are purchased. The following costs provide some guidelines.

- Enhanced color monitor	$2000
- High resolution color moniter	$9000
- Compilers (price per compiler)	$ 500
- GKS	$ 500
- UNIX	$ 500 - 1000

8 Bit Microcompter

Basic Version	-64 K RAM
	-360 K floppy disk drive
Upgraded Version	-640 K RAM
	-360 K floppy disk drive
	-20 mb hard disk drive

These machines range from a low of approximately $1,000.00 for the basic version to a high of approximately $8,000.00 for the upgrade.

The costs listed for the various computing powers are based on the information available at the time this report was prepared. Fluctuations in these costs occur rapidly as new technology is developed.

Computer maintenance costs are an important operating cost to be considered. These costs are also difficult to estimate. However, of the larger computer houses have developed guidelines for maintenance costs. These guidelines range from a low cost of 2% of the hardware purchase price per year. A good median value of 5% of the hardware purchase price should be used in most estimates. Higher maintenance costs are generally caused by misuse or poor environmental conditions.

The selection process involved in the purchase of a new microcomputer should include the following:

1) a detailed description at what the new computer will be used for
2) determination of the memory requirements for the new computer
3) a review of the hardware necessary to perform the desired tasks and a determination of the power of of computer required (8, 16, or 32 bit architecture)
4) peripheral equipment required and compatibility with the computers in the organization

Each organization should develop a global computer purchasing strategy that involves linking the necessary micro,

mini or mainframe computers together in order to share common data bases.

7.4 Input Devices

Computer input devices consist of apparatus used to encode or transform data into a digital format for use by the computer. Input devices include keyboards, light pens, mice, digitizing tablets, scanners and other light sensors.

Keyboards are the most common piece of input hardware. Keyboards consist of a series of push buttons (numbers, letters or symbols) that when pressed encode the command in machine understandable format. Keyboards may be similar to the very familiar typewriter keyboard. The operation of the keyboard is very well understood and for this reason keyboards are generally inexpensive.

Some keyboards have special function keys which upon depression activate an entire function rather than a single command at a time. Keyboards also have cursor control keys and other special keys.

There are two main alphabetic character layouts, the Querty and the Dvorak. The Querty key layout is the most common and has the format of the familiar typewriter. The Dvorak key layout was developed in 1932 by August Dvorak. The Dvorak key layout was designed with ease of operation and speed in mind. The key layout places the vowels and the most often used

consonants in the middle row, with the lesser used consonants above and below. The design minimizes finger movement and increases typing speed. This layout, however, has not been widely accepted due to the popularity of the Querty key layout. Some keyboards are available that may operate with either layout simply by changing the position of a switch.

Mice are devices used to replace the traditional cursor movement using cursor control keys. Two types of mice exist, the mechanical mouse and the optical mouse. The mechanical mouse controls cursor movement by monitoring position of a ball affixed to the bottom of the mouse. Movement of the mouse up and down or left and right causes the ball to rotate and thus the cursor to move in that particular direction. The optical mouse functions in a similar manner. The optical mouse is moved over a specially prepared co-ordinate pad. As the mouse moves over the pad, it optically detects its change in position and has the cursor move in the appropriate direction. The optical mouse lacks much of the mechanical nature of the mechanical mouse and thus its reliability has been shown to be substantially higher.

Other special functions of the mouse include the use of the mouse to indicate selection of options on the screen. Mice range in cost from approximately $50.00 to $300.00.

Touch technology has been advancing and computer screens have been developed that are touch sensitive. These screens have been subdivided into 4 grids defined by infrared light beams passing across the screen. By placing a finger at a

location of the screen, the computer can locate which light beams have been intersected, and thus the particular part of the grid the finger is pointing to. On the screen many different prompts may exist all of which are displayed at different co-ordinate locations. In this method many different prompts may be answered by pointing or touching a particular portion of the screen.

The light pen functions in a similar manner to the touch sensitive screens. However, it can pinpoint to a much greater accuracy due to the smaller size of the light pen and thus the capability of a much more refined screen grid. Light pens are often used to aid in computer aided design and are used to draw lines on the screen.

Scanners are rapidly achieving a fundamental role in inventory control and other areas that require recording of numerical or character type data. Scanners operate either using magnetic ink to assist in character recognition, or a bar code (common in the supermarket setting). Mining applications of the bar code type scanner include inventory control of both surface and underground supplies. On surface any articles in general warehouses can be controlled using bar codes. Underground, many supplies including drill steel and pipe fitting can be charged to specific cost centres directly through the computer. The use of the optical scanners can eliminate much of the paper work involved in inventory control. Optical wands and other required hardware, range in price from several hundred dollars to tens of thousands of dollars depending on the sophistication of the

system and the number of wands required. These systems range, from a computer the size of a pocket calculator linked to the optical wand that "dumps" the stored data to a larger computer systematically, to sophisticated stock control packages.

7.5 Output Systems

There are two basic types of computer output, (1) hardcopy referring to a recording on paper or film, or (2) soft copy referring to a temporary display such as a monitor.

Hardcopy output forms a permanent record, and is usually made by printers, camera systems and plotters. A wide variety of printer types of varying quality exist. These printers may be subdivided into two major groups, impact and non-impact printers.

The first group of impact printers is the dot matrix group. These printers form images by driving a series of small pins against the print ribbon. Dot matrix printers have two main advantages over other varieties; cost and flexibility. The cost is kept low due to the relatively simple technology. The flexibility arises from the ability to change the printer software and thus use the available print pins in any fashion desired.

The cost of a dot matrix printer varies.

Typical costs of printers are given in the following:

For personal computers	$ 400 - $2,000
For mini computers and Workstations	$1,000 - $3,000 for low duty cycle environment
	$4,000 - $8,000 for high duty cycle environment
	$8,000 - 60,000 for system printers

The varience in price is due to a difference in quality, speed and manufacturer of the printer. The quality of print from a dot matrix printer is directly related to the density and size of the print pins. The higher the density of the print pins and the smaller the print pins, the higher the quality of the printer. Print speed for the various printers can vary from 80 characters per second to 400 characters per second.

Letter quality printers are the second group of impact printers. These printers derive their names from their print being equal in quality to an electric typewriter. These types of printers may use one of several different techniques to impact the print ribbon with the desired character. Ball type elements, cylinders and daisy wheels all fall into this group. Printer speed is much dependent on the mechanical movement speed of the printer. The maximum speed of these printers is generally about 55 characters per second. Typical cost of letter quality printers are between $1,500 and $3,000.

The other variety of impact printers are those referred to as line printers. These printers are generally large mechanical types. Unlike the printers previously described these printers set up an entire line of print at a single time.

Thus, although an entire line of print is not typed at a single moment, these printers look as though this is the case. The characters used for imprinting are either of the dot matrix type or of the letter quality type. This type of printer is often very noisy and suffers from lack of print quality. Comparable quality dot matrix printers are available at competitive prices.

NON-IMPACT PRINTERS

As the name implies non-impact printers do not have a printhead or character string that actually hits the print paper. These non-impact print methods use nozzles, heat, electricity, or optical methods to make the image transfer.

The ink jet type printer is one of the non-impact printers. Printing is accomplished using a series of small jets of ink and functions in a similar manner to the dot matrix printer. The small dots or ink jets are in the pattern of characters. One major advantage of the ink jet printer is its color printing ability. Other advantages include its quietness and its high reliability due to the smaller number of moving parts as compared to other printer types.

Typical cost for ink jet printers are:

- for microcomputers	$ 2,000 - $ 3,000
- for minis and workstations	$15,000 - $25,000

Thermal and electrostatic printers are similar and therefore are classed together. Rather than using ink, these printers use specially treated papers. The thermal printer uses chemically treated paper sensitive to heat. The electrostatic printer uses an aluminum oxide treated paper sensitive to electrostatic charge. As the print head moves over the treated paper, the print head either heats the paper in the shape of the characters or sets up an electrostatic charge in the shape of the characters. Both types of printers suffer from the paper being more expensive and the printed characters fading with time.

Another type of thermal printer is the thermal transfer printer, in which heat is used in conjunction with an inked-waxed ribbon. The heat melts the wax on the ribbon and allows the ink to be transferred to the print paper. The thermal transfer printer uses ink; thus printed material does not fade as in the case of treated thermal paper.

The last and possibly the most important of the non-impact printers is the laser printer. Laser printers are similar in looks to a photocopier, however, the image is transferred from a computer rather than from an original copy. The output from the laser printers is of letter quality. Laser printers are expensive at the present time. Typical cost are between $4,000 and $8,000.

PLOTTERS

Plotters are the other main type of hardcopy output from computers. Plotters are devices that either move paper under fixed pens in the x-y directions or move pens over fixed paper in the x-y directions. Plotters are used for graphing, developing plan or cross sectional maps or any other accurate drawing. The computer sends the plotter x and y co-ordinate data in order to develop the desired pattern. Plotters are advancing technologically at tremendous pace and are capable of plotting in a variety of colors.

The costs of the plotters vary, depending on the size of the paper they can take, the number of pens they can use while drawing, and the speed at which they operate.

Typical prices are given in the following:

Size	Number of pens	Price
A	2	1,500 - 2,000
A	8	2,000 - 3,000
A and B	6	2,500 - 3,500
A and B	8	4,000 - 6,000
C and D	8	9,000 - 16,000
E	8	20,000 - 30,000

VIDEO DISPLAY TERMINAL OR CATHODE RAY TUBES (CRT)

Cathode ray tubes have many applications, the most common of which is probably the ordinary television screen. These screens work on the principle of an electron beam that scans across a phosphor coated surface. This scanning leaves

points of light behind on the tube. Brighter areas are made by increasing the intensity of the electron beam at that particular area of the scan and darker areas are made vice-versa. Each scan line is made of many points called pixels. The intensity of each pixel can be varied independently of the others. These pixels are defined by row and column numbers.

Typical screen sizes are 24 rows by 80 columns wide. Variations to this format do appear however. The rows and columns are further divided into pixels, a typical screen being 200 by 360 pixels wide. Thus, a typical screen for a personal computer is composed of 72,000 pixels.

Both monochrome and color displays are available with the color display two to five times the cost of the same quality monochrome display. Color CRT's are developed using three separate (red, green, blue (RGB)) electron beams. A large variety of displays are available, the largest single difference between them being the number of pixels the screen is composed of, and thus the quality or resolution of the screen.

7.6 Modems

A modem is a device used to allow computers to communicate via telephone or similar lines. Telephone lines have been designed for analog communication. Computers are digital machines that communicate via a series of discrete on/off signals. Thus, in order for the digital computer to send signals

via the telephone lines, a device (modem) is required at the sending end to convert the computer's digital signal to an analog signal. A device (modem) is also needed at the receiving end to convert the analog signal to a digital signal. Many types of modems exist. The earliest and most common form is the acoustic modem. It consists of an acoustic coupler, designed to hold the hand set of an ordinary telephone, and modem circuits to convert the signals and to interface with a terminal or personal computer. With the advent of the standard telephone "jack" connection, newer modems have taken on a slightly different form. They consist of the interface circuits along with a standard plug that the telephone jack can fit into.

Transmission capacity of modems is measured in baud. This represents the number of bits of information which are communicated in 1 second. Typical baud rates range from 110 - 300 to 4800 or even 9600 baud.

Modem costs range from under $200.00 to over $2,000.00, with a typical cost of about $500. The reason for the large variation in price is due to a wide discrepancy between the capabilities of the various modems. The most typical modem is the one which places the phone handle in a cradle. The most sophisticated modems are referred to as intelligent modems and are programmed to automatically perform dialing, answering, and disconnection functions.

CHAPTER 8

SPECIFICATIONS FOR THE CREATION OF A SOFTWARE LIBRARY FOR THE ONTARIO MINING INDUSTRY

8.0 INTRODUCTION

In order to maximize the usage of available computer software for the Mining Industry it is desirable that a library of computer software is established. This library should "store" computer programs and information which could be available to its members. The following is a discussion on the concept of the Mining Software Library and provides specifications for its creation.

8.1 Mining Software Computer Library

The Mining Software Library should serve the following purposes:

1. Information on existing software.
2. Distribution of software.
3. Maintenance of software.

The first step towards the creation of a Library should be the collection of available software and literature related to

it. Since the already existing programs may not follow the exact software engineering and other specifications set by the present report, it is recommended that each program is tested, evaluated and a report is prepared. It is recommended that a copy of the program is available for the use of the Library staff which will be bound by legal secrecy agreements towards the agency having the copywrite of the code. If this is not possible, the staff of the Library will evaluate the product at any site which can provide a working up-to-date version of the code. It will be the responsibility of the maintainers of the code to inform the Library with regard to new versions.

In the case of non restricted software, the Library will be able to distribute and maintain it.

The Library will produce the following report for each product.

1. Abstract Report

The format of the abstract report is shown in table 13. This report will become available on request.

2. Evaluation Report

The evaluation report will provide a general description of the code, the data used, the results obtained and the versions tested during the evaluation period of the code. This will be available to the members of the Library.

1. NAME AND DESTINATION OF THE PROGRAM.
2. COMPUTER FOR WHICH PROGRAM IS DESIGNED AND OTHER MACHINE VERSION PACKAGES AVAILABLE.
3. DESCRIPTION OF PROBLEM OR FUNCTION.
4. METHOD OF SOLUTION.
5. RESTRICTIONS ON THE COMPLEXITY OF THE PROBLEM.
6. TYPICAL RUNNING TIME.
7. TYPICAL PRIMARY AND SECONDARY MEMORY REQUIREMENTS (MACHINE REQUIREMENTS).
8. UNUSUAL FEATURES OF THE PROGRAM.
9. PROGRAMMING LANGUAGES USED.
10. OPERATING SYSTEM UNDER WHICH PROGRAM IS EXECUTED.
11. PROGRAMMING OR OPERATING INFORMATION OR RESTRICTIONS.
12. STATUS.
13. AUTHOR(S) - AGENCY/CONTRACTOR/SPONSOR.
14. AVAILABILITY.
15. MATERIAL AVAILABLE.
16. SUPPORT BY THE AUTHOR.
17. SUPPORT BY THE LIBRARY.
18. MATERIAL AVAILABLE.
19. RELATED AND AUXILIARY PROGRAMS.

TABLE 13: FORMAT OF ABSTRACT REPORT.

3. Maintenance Report

This will be a report on the maintenance of the program. It will be sent to the users of the program for programs distributed by the Library.

Cost and Membership Issues

The initial effort to create the Library will require the services of two - three engineers for a period of two years. After the initial period, the Library should be able to be self supported. In order to do this, a membership fee will be required. Moreover a price should be set for each software product and each product report available through the Library. The following membership policy is suggested:

- Canadian Mining Companies provide an initial fee which will give them the right to obtain evaluation reports and software for as long as the funds last. Upon exhaustion of the funds the fee should be renewed.

- Canadian Universities will have access to the Library at no cost. It is expected that the Universities will cooperate with the Library.

Companies and Universities outside of Canada will be able to become members of the Library at an increased fee.

CHAPTER 9

CONCLUSIONS

Software engineering principles concerning the production and maintenance of software products for the use of the mining industry have been developed. It is realized that a software product developed for the mining industry has many users and maintainers. Therefore, it is important that the products are developed on the concepts of ease of maintenance and use. A software product consists of several deliverable items in addition to the source code. Typical deliverables include a user's manual, the design phase document and the theoretical description document. Specifications were set for each of these documents.

Standard operating systems were examined for use by mining companies. UNIX was selected as a standard because of its portability, multi-user, multi-tasking capabilities and its memory addressability.

A large variety of computer languages have been examined in order to select the most appropriate one for mining engineering applications. The programming needs of the mining industry are extensive and include database and text processing programs, traditional numerical processing programs and low level

programming for such activities as robotics and machine control. Five languages (BASIC, C, FORTRAN, PASCAL and PL/1), were evaluated. The C language was selected because of its multipurpose capabilities. C is an excellent general purpose language and its programming style leads to portable computer programs.

Graphics software and hardware for use by the mining industry have been reviewed in considerable detail. For two dimensional graphics the Graphical Kernel System (GKS) has been internationally accepted as a standard. It is recommended that the Ontario Mining Industry adhere to this policy. Three dimensional graphics software on the other hand has not been internationally standardized as of yet and it may be several years before a standard is set. For this reason several three dimensional packages have been examined.

Database management systems have been reviewed. Problems associated with databases and advantages from their use were presented and the applications associated with the Mining Industry were indicated.

Microcomputer hardware has been examined. This consisted of a review of the functioning of the microcomputer with emphasis placed on the understanding of microcomputer specifications. Computer hardware including input/output devices, memory and modems were examined. Rule of thumb costs were developed for a range of microcomputer systems and support hardware for computer maintenance.

Finally a library of mining related software was proposed. This library will collect information on existing software and distribute and maintain software products. It will also produce evaluation reports and complete abstracts for available software.

REFERENCES

1. Fairley, R.: "Software Engineering Concepts", McGraw Hill, 1985.

2. Peterson, J.L., Silberschatz, A.: "Operating System Concepts", Addison-Wesley, 1985.

3. Weinberg, P.N., Groff, J.R.: "Understanding Xenix", Que Corporation, Indianapolis 1985.

4. Kelley, A., Pohl, E.: "A Book on C", Benjamin/Cummings Publishing Co., 1984.

5. Harrison, S.P., Steele, G.L.: "C - A Reference Manual", Tartan Laboratories, Prentice Hall, 1984.

6. Gehani, N.: "C: An Advanced Introduction", Computer Science Press, Bell Telephone Laboratories, 1985.

7. Wagener, J.: "FORTRAN 77 - Principles of Programming", John Wiley & Sons, 1980.

8. IBM: "IBM Personal Computer - XENIX" 1985.

9. BYTE Magazine: Issues May 1985 - November 1986.

10. UNIX World Magazine: Issues May 1985 - October 1985.

11. Computer Data Magazine: Issues March 1986 - November 1986.

12. Smith, P.D., Barnes, G.M.: "Files and Databases on Introduction", Addison-Wesley, 1978.